EFI 2.0 Specification

AUTHORS: EWOUD WERKMAN, MENTE KONSMAN, WILCO WIJBRANDI, BOB RAN

Table of Contents

1 Introduction

This document explains the XML version of the Energy Flexibility Interface (EFI). The corresponding XSD can be found at GitHub:

https://github.com/flexiblepower/efi/blob/master/schema/efi-2.0.xsd

1.1 What is energy flexibility and why is it important?

Smart grids rely on flexibility in energy production and/or consumption to compensate for the increasing numbers of renewable energy sources that are far less predictable/controllable than traditional power plants. The required flexibility ultimately comes from smart devices in households, SME's, office buildings, etc.

Energy flexibility can be defined as the ability to willingly deviate from the normal energy production and/or consumption pattern over time and/or by power level. This flexibility may be used by third parties to help alleviate imbalance or congestion.

Third parties will use different incentive schemes to unlock the flexibility potential, such as time of day pricing, real time pricing and feed in tariffs. These incentives should somehow be mapped to the possibilities of smart devices to deliver energy flexibility.

1.1.1 What is EFI?

As of yet there is no standard interface to describe and control the energy flexibility of smart devices. The Energy Flexibility Interface, for short EFI, fills this gap and is specifically designed as a standard communication method between smart devices and Demand-Side Management (DSM) solutions. EFI is developed by TNO to deal with interoperability issues encountered while experimenting and researching energy flexibility in field trials. It is a key enabling technology for the widespread deployment and adoption of Demand-Site Management to exploit energy flexibility. The interface specification of EFI is open and freely available.

1.1.2 Why EFI?

Currently, there are many Demand-Side Management (DSM) solutions available that exploit the flexibility of energy devices, such as USEF, OpenADR, Triana and TNO's own PowerMatcher and HeatMatcher. All of these solutions initially developed their own way to model the flexibility and communicate with the managed energy device, resulting in many different protocols. Since there are many devices, this doesn't scale well: device vendors need to support many DSM solutions and DSM vendors need to support all possible devices. Furthermore, buying one (silo) solution will create a vendor lock-in, restricting consumers in freely choosing their DSM and device combinations. This limits the speed of adoption of exploiting flexibility, delaying the energy transition.

EFI objectives:

- interoperability between demand response services and smart devices
- open market
  - lower barriers for new service providers/aggregators
  - accelerating innovation by preventing DSM silo solutions / vendor lock-in.
  - flexibility providers should be able to easily switch from one service provider/aggregator to another
- provide a solid base for future developments of DSM solutions and/or smart devices
- simplify architectures of smart grid solutions by separation of concerns

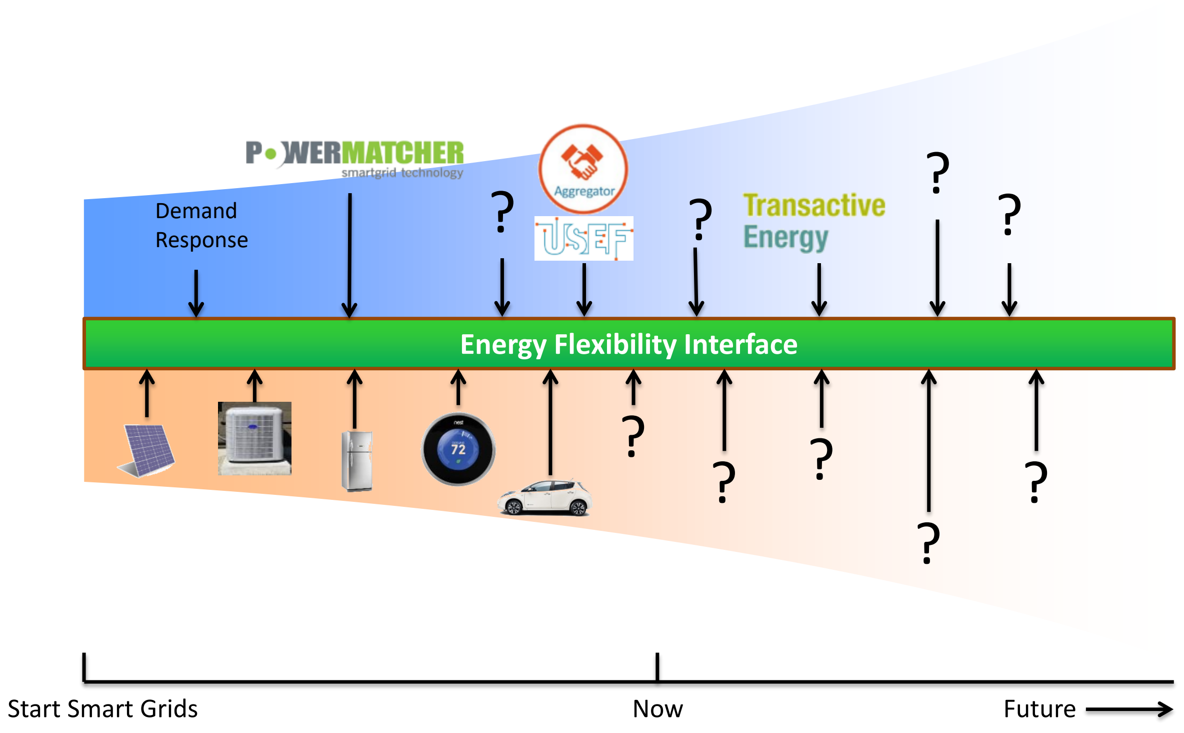

FIGURE 1. *FLEXIBILITY IS THE ONLY CONSTANT* IN THE DEVELOPMENT OF SMART GRIDS. WE DON'T KNOW WHAT DSM SOLUTIONS WILL APPEAR NOR WHAT SMART DEVICES WILL BECOME AVAILABLE. WHAT WE DO KNOW, IS THAT WE WANT TO EXPLOIT THEIR FLEXIBILITY IN A SIMPLE AND UNIFORM WAY.

1.2 How does EFI work?

EFI introduces interoperability in the communication between device and DSM. TNO has thoroughly analysed the information exchange between device and DSM and has defined four different device categories to model energy flexibility. They provide an abstraction of the devices regarding energy flexibility and are device and DSM independent. This document explains how EFI works.

1.2.1 EFI as a common language for energy flexibility

EFI creates a common language for energy flexibility (as depicted in Figure 2), allowing any combination of device and DSM-solution. It is designed to be future proof and only provides an abstraction of the device for modelling energy flexibility (e.g. it does not solve issues regarding home automation).

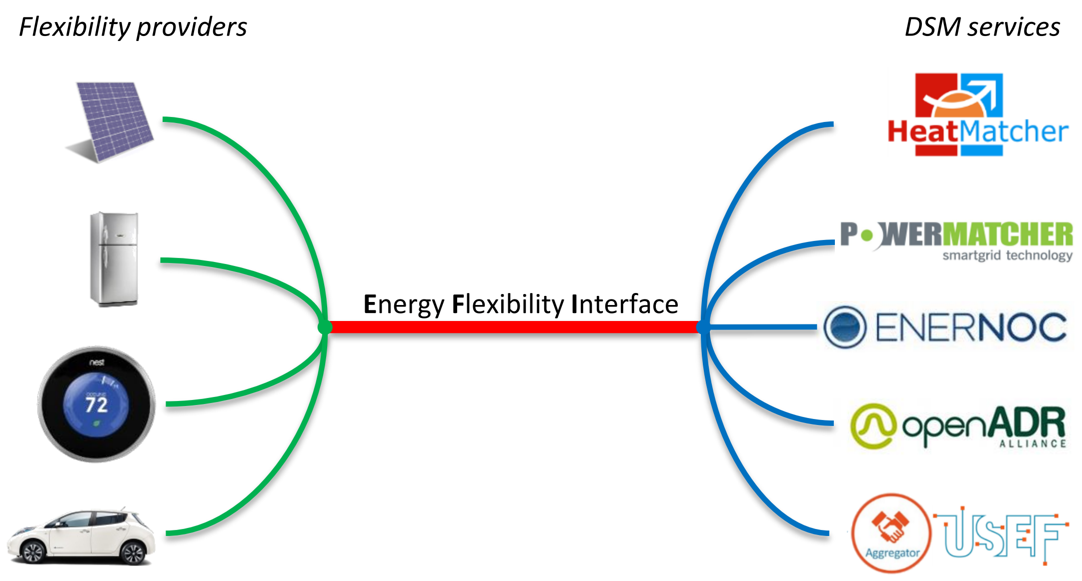

FIGURE 2. EFI PROVIDES ONE COMMON LANGUAGE FOR ENERGY FLEXIBILITY. THIS ALLOWS ALL DEVICES TO COMMUNICATE WITH ALL DEMAND-SIDE MANAGEMENT SOLUTIONS WITHOUT HAVING TO DEVELOP CUSTOM ADAPTERS FOR EACH COMBINATION. IT CREATES AN OPEN PLAYING FIELD FOR ALL PARTIES REGARDING FLEXIBILITY SERVICES.

1.2.2 Flexibility Interface for Aggregators

Besides device energy flexibility, higher level aggregated flexibility (such as flexibility of cold stores, multiple households and industrial processes) can be modelled using EFI with the Aggregator category of EFI. This category is an extended version of the Shiftable category and supports flexibility models required for trading on day-ahead, intra-day, and real-time energy markets by aggregators.

1.3 Governance by the Flexible Power Alliance Network (FAN)

EFI is available as open source standard which is currently governed by the FAN foundation, see http:// http://flexible-energy.eu/ for more information. Currently, EFI concepts are being standardized within the European standardization organization CENELEC, to become a European Standard for modelling energy flexibility.

2 Architecture and Common Data Elements

2.1 Architecture

The architecture of EFI, depicted in the figure below, shows the main concepts around EFI. EFI is centred in the middle, as the common language between the Smart Grid and the Device. The demand-side management solution is represented by the Energy App. The device is represented by the Device Driver. This Device Driver is able to monitor and control the physical device. Most of the time there is a specific communication protocol to communicate with the device, such as Zigbee, PowerLine Communications (PLC) and WiFi. The figure shows the scope of TNOs reference implementation of EFI and supporting execution platform, called Energy Flexibility Platform & Interface (EFPI)[1], too.

2.1.1 Scope

Creating and destroying communication channels is out-of-scope of the EFI specification; EFI is designed such that it can run over any bidirectional communication channel, for example XMPP or Websockets. For more information, refer to EFPI[2], as it contains a reference implementation of EFI including a ConnectionManager that takes care of the communication channel life-cycle.

[1] EFPI can be downloaded for free from http://flexible-energy.eu/

[2] For more information about EFPI see http://flexible-energy.eu/

Present day solutions provide some sort of model of the device they control, but never in general terms. Thus these solutions must model each devices they wish to support. However, modelling a single device might not be sufficient as there are often many manufacturers of the same type of device. Depending on the level of abstraction used, a technology must implement one or more models.

The level of abstraction can be taken one step further by not modelling devices, but rather the nature of it. It is not the device that is modelled but rather its energy flexibility. From this a real device can be connected and used to perform the communication between the derived class and its device. In our experience four models are sufficient enough to cover all device types. These models are called flexibility categories and describe a so called Control Spaces of that category, i.e. a Device sends its Control Space to the energy app, by using one of the flexibility categories of EFI.

In essence, a control space is a way to put the information that is contained within a device into a generic structure, such that Energy App are able to understand that device from a generic energy model. Where Control Spaces form an abstract representation of a device, Allocations are used to express what a device is requested to do by the Energy App. For each Control Space, there is also one Allocation type.

2.2 Flexibility categories

The Energy Flexibility Interface (EFI) consists of four flexibility categories.

Flexibility category	***Description***	***Examples***
Inflexible	Cannot be controlled and has no actual flexibility, but is measurable and may provide forecasts	*Photo-voltaic panels, domestic loads, windmills, solar collectors*
Shiftable	Process which can be shifted in time, e.g. has a deadline	*Washing machines, dryers, dishwashers*
Storage	Flexible in production / consumption level, but is bounded by a buffer. Deadlines and required fill levels constrain the flexibility of this category.	*Freezers, CHPs, thermal buffers, stationary batteries, electrical vehicles*
Adjustable	Flexible in production / consumption level and not constrained by a buffer. They have a wide range of control possibilities without many restrictions and therefore usually offers a lot of flexibility.	*Generators, dimmable lighting, heat pumps, gas boilers*

2.2.1 Inflexible

Inflexible control spaces are those that cannot be controlled and therefore cannot be used in a flexible manner. This applies in particular to the renewable energy sources, such as PV panels and wind turbines. But also to inflexible part of domestic loads, i.e. TV set, indoor lighting, coffee machine, etc. Renewable energy sources depend on conditions in nature for their energy production. The amount of solar radiation determines the energy production for PV and the wind force does the same for wind turbines. These elements of nature cannot be curtailed and hence the Inflexible Control Space.

Within a domestic environment, there are several components which realistically cannot be controlled. One would not like to be denied access to one’s TV set if congestion management demands for less energy. The same hold true with coffee machines, water cookers, etc. The end users determines the usage and not the system. Hence the Inflexible Control Space.

Although most inflexible devices are exactly what their name implies, there are some instances of inflexible devices that actually can be controlled in a limited way. E.g. a PV inverter might be controlled to curtail its power output so that it will never produce more than a certain predefined limit. This can be useful to prevent peaks on the electricity grid. In that case the allocation will communicate the maximum amount of production or consumption that an inflexible device is allowed to.

2.2.2 Shiftable

Where the Inflexible Control Space cannot be controlled, the Shiftable Control Space can. The usage patterns of Shiftable devices allows for usage shifting in time. It is not about using an device less, but rather using it at different moments in time. Examples of such devices are washing machines and dishwashers. Both need to run at some time to either clean clothing or dishes. The exact time at which this is done is not that important. This method introduces flexibility in the electrical system by moving the usage of electrical energy in time. If the electrical system needs less consumption, the running of these devices can postponed to be better suiting moment. Once the device starts working, it performs its normal operation and cannot be controlled anymore.

Another characteristic of this class of devices is that there is a need to be finished before a fixed time. The dished need to be clean and dry before breakfast next morning. This deadline must be adhered to, even if it goes against the needs of the electrical system.

The flexibility of this category consists of shifting the start time to a moment that is most optimal. Therefore the allocation consists of simply sending the desired start time for this device.

2.2.3 Storage

Some devices have flexibility to temporarily store energy in some form of a buffer (e.g. a water buffer or a battery). The Control Space lists the flexibility in either production or consumption of energy that is available between the minimum and maximum fill level of the buffer. The usage is not shifted in time, rather its usage is altered in time to provide an optimum solution.

Energy can be stored electrically, in which case it is a battery. Another example of energy storage is storing it as heat or cold. A refrigerator or freezer stores cold and when it warms up, one can say the buffer is slowly emptied. At a certain predefined point, i.e. temperature, the refrigerator or freezer will turn on and start cooling again, i.e. filling the buffer again. This filling does not need to be done at exactly the same temperature each time. It can be done a little sooner or later, depending on the current state of the electrical system. Of course, if the temperature rises too high, then there is no flexibility anymore and the device must be turned on. For storing heat, either a heat pump or a Combined Heat and Power (CHP) in combination with a water buffer is used. The water is heated and used for space heating or tap water. The state of the water buffer determines the amount flexibility that is offered. The difference between the two devices is that heat pumps use electrical energy for producing heat, while an CHP uses gas to generate heat and produces electricity. A CHP is a multi-commodity device, because it uses gas and produces electricity.

Just as heat and cold are stored in buffers, so is electrical energy in a battery. It too fits within the Storage Control Space and is governed much the same as with heat and cold. The Electrical Vehicle (EV) is not much different from a battery when it is used in Vehicle to Grid (V2G) scenarios. The exception is that where a battery is usually stationary, the EV will change location and its State Of Charge (SoC) can change unpredictably when it reconnects to the electrical system.

A storage device has multiple modes that it can be in, e.g.: switched on, switched off, different modulations. The allocation states which mode the device has to switch to, and at what time.

2.2.4 Adjustable

All discussed control spaces thus far have been constrained in one way or another. The Adjustable Control Spaces have no such restrictions, since it can be adjusted at will. An example of Adjustable devices are diesel generators or gas boilers. The Control Space lists the flexibility in either production or consumption of energy without being bound to a buffer or deadline.

Adjustable Control Spaces are much like Buffer or Storage Control Spaces, but without a buffer. A generator is such an example. It can be switched on in order to provide extra power to the electrical system and can be switched off when the need has dissipated. There are only technical device constraints when it can or cannot run, such as the amount of diesel it has left, how fast the generator may be started again after it has been stopped, etc.

The allocation of Adjustable devices is similar to that of buffer and storage devices.

2.3 Which category to use?

The previous section described the four different types of Control Spaces. A question that might arise is on how to decide what Control Space is suitable for which appliance or device? This will lead to a decision tree as depicted in Figure 2. The first question that must be answered about the device is: has it any form of flexibility in it? If it does not, from the flexibility point of view, there is no desire to control it and hence the Inflexible Control Space.

In case the appliance does have flexibility to offer, it can do so in different forms. The first one is if the usage of the appliance can be shifted in time and there is a deadline before the device must be finished. If it can and does, the Shiftable Control Space is appropriate. You do not change the behaviour of the appliance, rather you change the moment at which the appliance is switch on and off.

When the full operation of the device cannot be shifted in time, then it might be that there are constrains on the device, such as a buffer. Asking the question if the appliance is freely usable, yields the Buffer Control Space in case it is not and if the answer is yes, yields in the Adjustable Control Space.

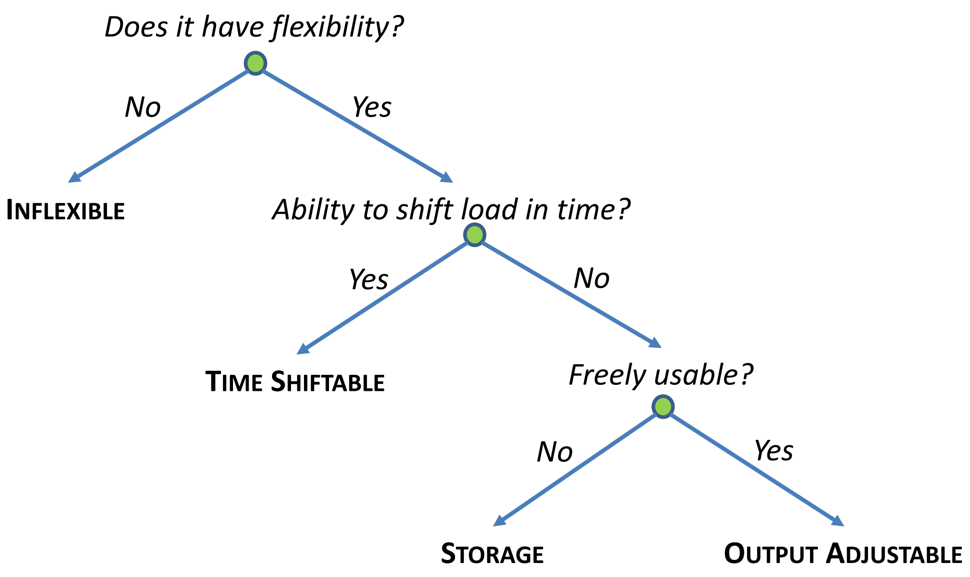

FIGURE 3: DECISION TREE FOR DETERMINING WHICH CLASS OF CONTROL SPACE TO USE FOR WHICH DEVICE.

3 Common S2 Data Elements

The S2 interface discerns four distinctive flexibility categories. These categories share a number of common messages and data elements. These elements are introduced in this chapter, the subsequent chapters will go into the specifics of each category.

3.1 Typical S2 message flow

In Figure 4 an overview is given of the general messages between the resource manager and the CEM. In the next paragraphs each message is explained in detail.

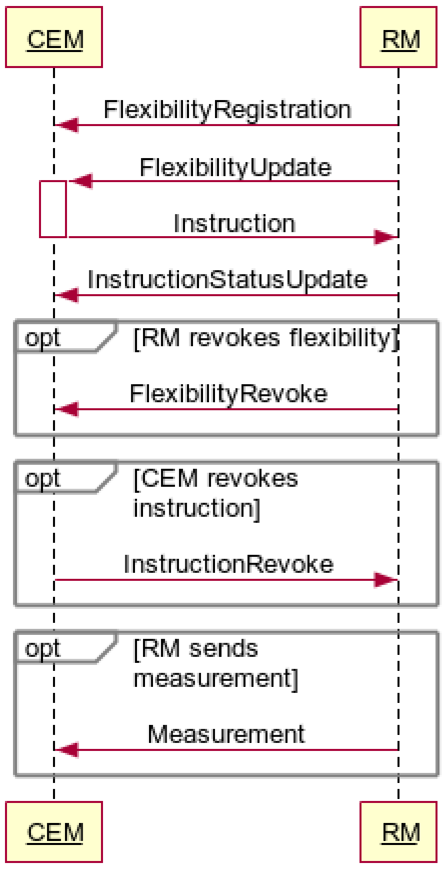

FIGURE 4: A TYPICAL S2 MESSAGE FLOW

3.2 Efi Message

This is a parent class from which all, more specific, messages have been derived.

Figure 2 shows the structure of the EfiMessage class.

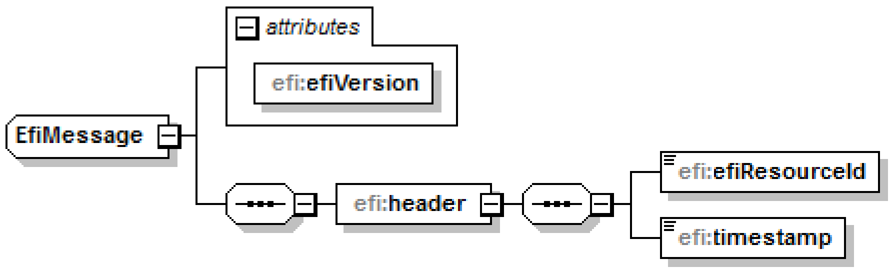

FIGURE 5: OVERVIEW OF THE EFIMESSAGE CLASS

The EfiMessage class features the following attributes:

Attribute	Description
efiVersion (string)	This attribute of the type string indicates the EFI version of this message. The version described in this document is

	“2.0”. The value of this attribute is therefore fixed to the string “2.0”.
efiResourceId (Identifier)	An ID that uniquely identifies the Resource Manager that this message contains information about.
Timestamp (dateTime)	This timestamp of the typedateTime indicates the moment in time this message was constructed.

TABLE 1: ATTRIBUTES OF THE EFIMESSAGE CLASS

3.3 FlexibilityRegistration

As soon as a resource manager becomes available to the CEM, it will send a message that is derived from this FlexibilityRegistration class. This message is used to inform the CEM about the capabilities of the resource.

Each of the flexibility categories has its own FlexibilityRegistration message derived from this class. Attributes about specific capabilities are added in the derived messages and are defined in the next chapters.

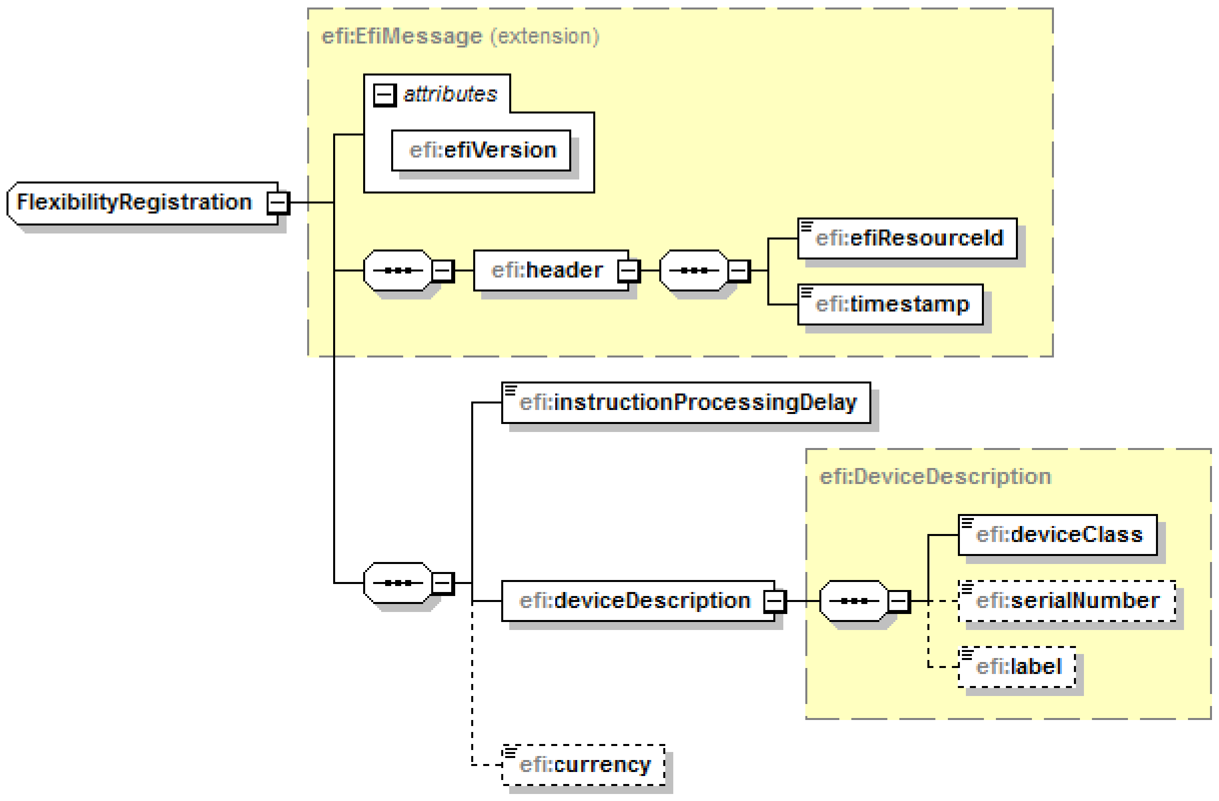

FIGURE 6: OVERVIEW OF THE FLEXIBILITYREGISTRATION CLASS

The FlexibilityRegistration class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
InstructionProcessingDelay (duration)	In response to a FlexbilityUpdate (see 3.4) from the Resource Manager an Instruction (see 3.5) may be sent by

	the CEM. It could very well be that the processing of the Instruction requires a certain period of time. It is this period that is indicated by this InstructionProcessingDelay attribute.
deviceClass (DeviceClass)	The deviceClass is an informational attribute that indicates the type of device that the Resource Manager is managing. It is an enumeration of possible device classes that contains the following values: • Refrigerator • Freezer • Water Cooler • Water Heater • Washing Machine • Clothes Dryer • Combo Washer Dryer • Drying Cabinet • Dishwasher • Heatpump • Micro-CHP • Stationary Battery • Electrical Vehicle • PV Panel • Windmill • Solar Collector • Air Conditioner • Ventilation • Air Quality Appliance • Gas Geater • Floor Heating • Generator • Miscellaneous
serialNumber (string)	This optional attribute can be used to communicate the serial number of the device the Resource Manager is managing.
label (string)	The optional label attribute can be used to further identify the device. This is a free format label and could

	for instance be used to provide the manufacturer and the typenumber of the device.
currency (CurrencyType)	In some cases depreciation costs can be associated with certain actions that are performed by the Resource Manager. If the Resource Manager chooses to communicate such costs in subsequent messages it can indicate which currency will be used by using this attribute. CurrencyType is an enumeration of all available currencies. • AED • ANG • AUD • CHE • CHF • CHW • EUR • GBP • LBP • LKR • LRD • LSL • LYD • MAD • MDL • MGA • MKD • MMK • MNT • MOP • MRO • MUR • MVR • MWK • MXN • MXV

- MYR
- MZN
- NAD
- NGN
- NIO
- NOK
- NPR
- NZD
- OMR
- PAB
- PEN
- PGK
- PHP
- PKR
- PLN
- PYG
- QAR
- RON
- RSD
- RUB
- RWF
- SAR
- SBD
- SCR
- SDG
- SEK
- SGD
- SHP
- SLL
- SOS
- SRD
- SSP

- STD
- SYP
- SZL
- THB
- TJS
- TMT
- TND
- TOP
- TRY
- TTD
- TWD
- TZS
- UAH
- UGX
- USD
- USN
- UYI
- UYU
- UZS
- VEF
- VND
- VUV
- WST
- XAG
- XAU
- XBA
- XBB
- XBC
- XBD
- XCD
- XOF
- XPD

	• XPF • XPT • XSU • XTS • XUA • XXX • YER • ZAR • ZMW • ZWL

TABLE 2: ATTRIBUTES OF THE FLEXIBILITYREGISTRATION CLASS

3.4 FlexibilityUpdate

After the registration of the Resource Manager with the CEM, the Resource Manage can now start sending its current flexibility options. Every flexibility category has its specific FlexibilityUpdate messages that are derived from this class.

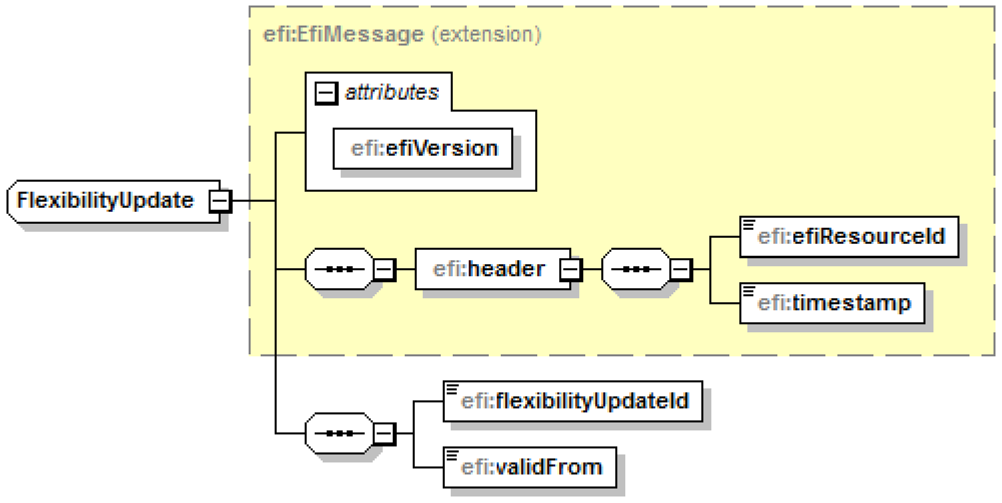

FIGURE 7: OVERVIEW OF THE FLEXIBILITYUPDATE CLASS

The FlexibilityUpdate class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
flexibilityUpdateId (Identifier)	An ID that uniquely identifies this FlexibilityUpdate.
validFrom (dateTime)	This timestamp indicates from which moment on this update is valid.

TABLE 3: ATTRIBUTES OF THE FLEXIBILITYUPDATE CLASS

3.5 Instruction

An Instruction message is always a response from the CEM to a FlexibilityUpdate message. It contains instructions on how to use (or fix) the flexibility described in the FlexibilityUpdate.

Each flexibility category has its own Instruction message that is derived from this one.

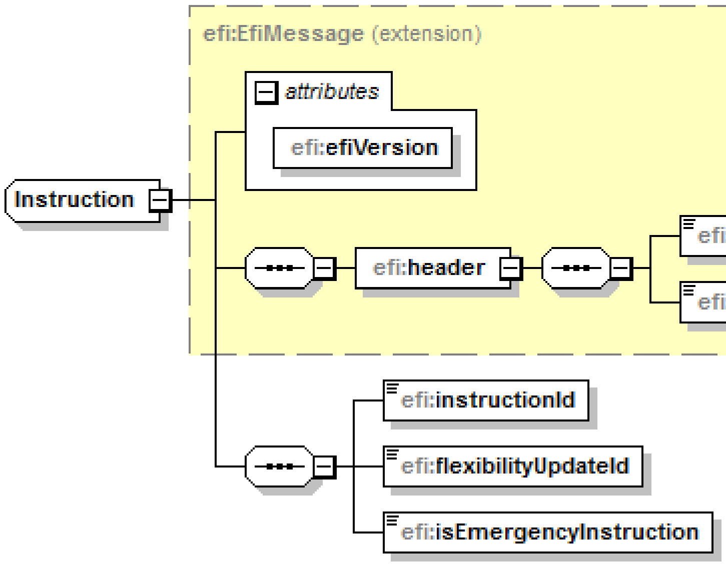

FIGURE 8: OVERVIEW OF THE INSTRUCTION CLASS

The Instruction class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
InstructionId (Identifier)	An ID that uniquely identifies this Instruction.
flexibilityUpdateId (Identifier)	An identifier that uniquely identifies the FlexibilityUpdate message that this message is a response to.
isEmergencyInstruction (boolean)	This attribute is true when a grid emergency situation occurs. (e.g. congestion, black start etc.) The CEM then strongly advices the Resource Manager to follow this Instruction in order to maintain grid stability.

TABLE 4: ATTRIBUTES OF THE INSTRUCTION CLASS

3.6 InstructionStatusUpdate

After a Resource Manager has received an Instruction message, it is good practice to provide feedback to the CEM about the follow up actions. This can be done via this message. Multiple InstructionStatusUpdate messages may be sent in response to a single Instruction.

All flexibility categories use this InstructionStatusUpdate class; there are no specific derivations in use.

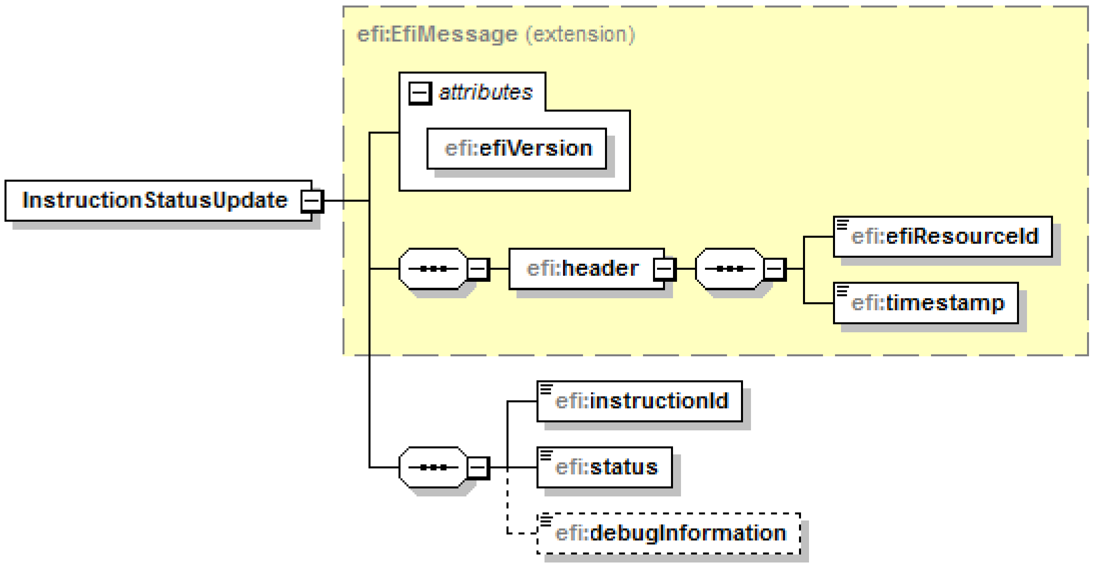

FIGURE 9: OVERVIEW OF THE INSTRUCTIONSTATUSUPDATE CLASS

The InstructionStatusUpdate class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
InstructionId (Identifier)	An ID that uniquely identifies the Instruction that this InstructionStatusUpdate provides feedback about.
status (InstructionStatus)	This attribute indicates the current execution status of the Instruction by the Resource Manager. It assumes one of the values specified in the InstructionStatus enumeration (see Figure 10 for more information): • ACCEPTED • STARTED • SUCCEEDED • REJECTED • ABORTED
debugInformation (string)	A developer of a Resource Manager may use this optional attribute to provide additional debug information. This is especially useful for the “REJECTED” and “ABORTED” status.

The following diagram explains which status transitions are allowed.

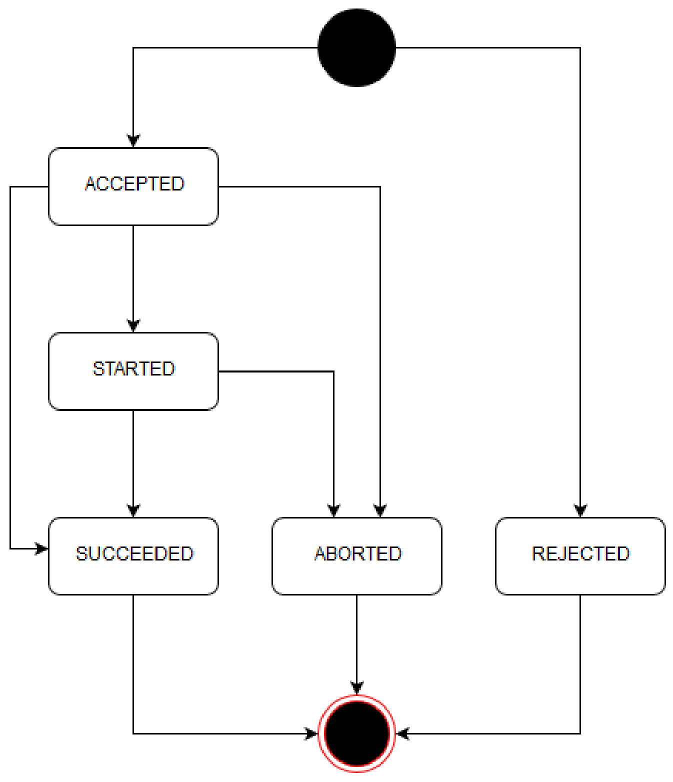

FIGURE 10: ALLOWED STATUS TRANSITIONS

An Instruction message is either “ACCEPTED” or “REJECTED”. When it is “REJECTED” no more InstructionStatusUpdate messages will follow. From the “ACCEPTED” state a transition can be made to “SUCCEEDED”, “STARTED” or “ABORTED”. The “STARTED” state is used when the execution of the Instruction is not merely an event, but takes time, as in the case of a power profile that has to be followed over time. From the “STARTED” state a transition to “SUCCEEDED” and “ABORTED” can be made.

3.7 FlexibilityRevoke

A Resource Manager may revoke FlexibilityUpdate messages that it already sent to the CEM. After sending this FlexibiltyRevoke message all FlexibilityUpdate messages that were sent before it are rendered invalid. The FlexibilityRegistration message does remain valid.

All flexibility categories use this FlexibilityRevoke class; there are no specific derivations in use.

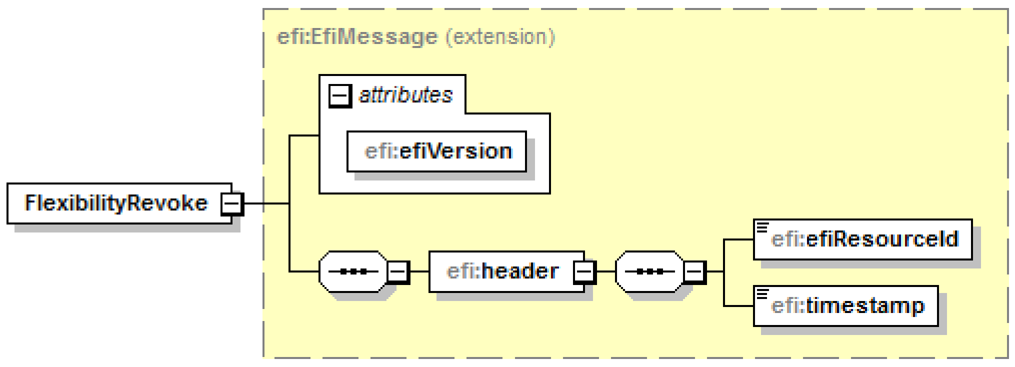

FIGURE 11: OVERVIEW OF THE FLEXIBILITYREVOKE CLASS

The FlexibilityRevoke class is derived from EfiMessage (see section 3.2) and does not feature additional attributes.

3.8 InstructionRevoke

The CEM may revoke an Instruction it sent earlier. In that case it will send this InstructionRevoke message.

All flexibility categories use this InstructionRevoke class; there are no specific derivations in use.

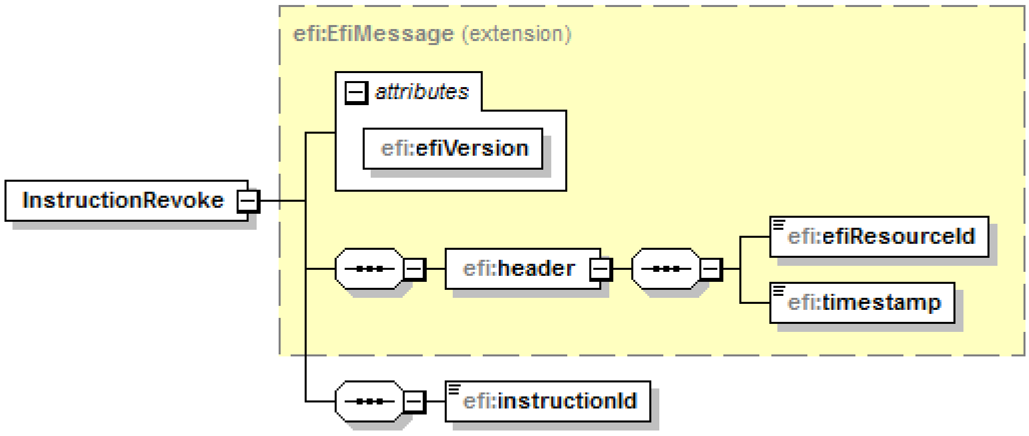

FIGURE 12: OVERVIEW OF THE FLEXIBILITYREVOKE CLASS

The InstructionRevoke class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
InstructionId (Identifier)	An ID that uniquely identifies the Instruction that is revoked by this InstructionRevoke message.

TABLE 5: ATTRIBUTES OF THE INSTRUCTIONREVOKE CLASS

3.9 Measurement

In principal the CEM knows what a Resource Manager is doing based on FlexibilityUpdate and InstructionStatusUpdate messages. In reality however the behaviour of the Resource Manager may deviate from the information it sent in these messages. This can have different causes, such as manual user intervention of the device managed by the Resource Manager or a lack of precision in forecasting information.

If available a Resource Manager may send Measurement messages to the CEM, so that the CEM is informed of what is actually happening and can determine whether there are significant differences to what it expected.

A Measurement message must contain one or more of the following measurements: electricityMeasurement, gasMeasurement or heatMeasurement.

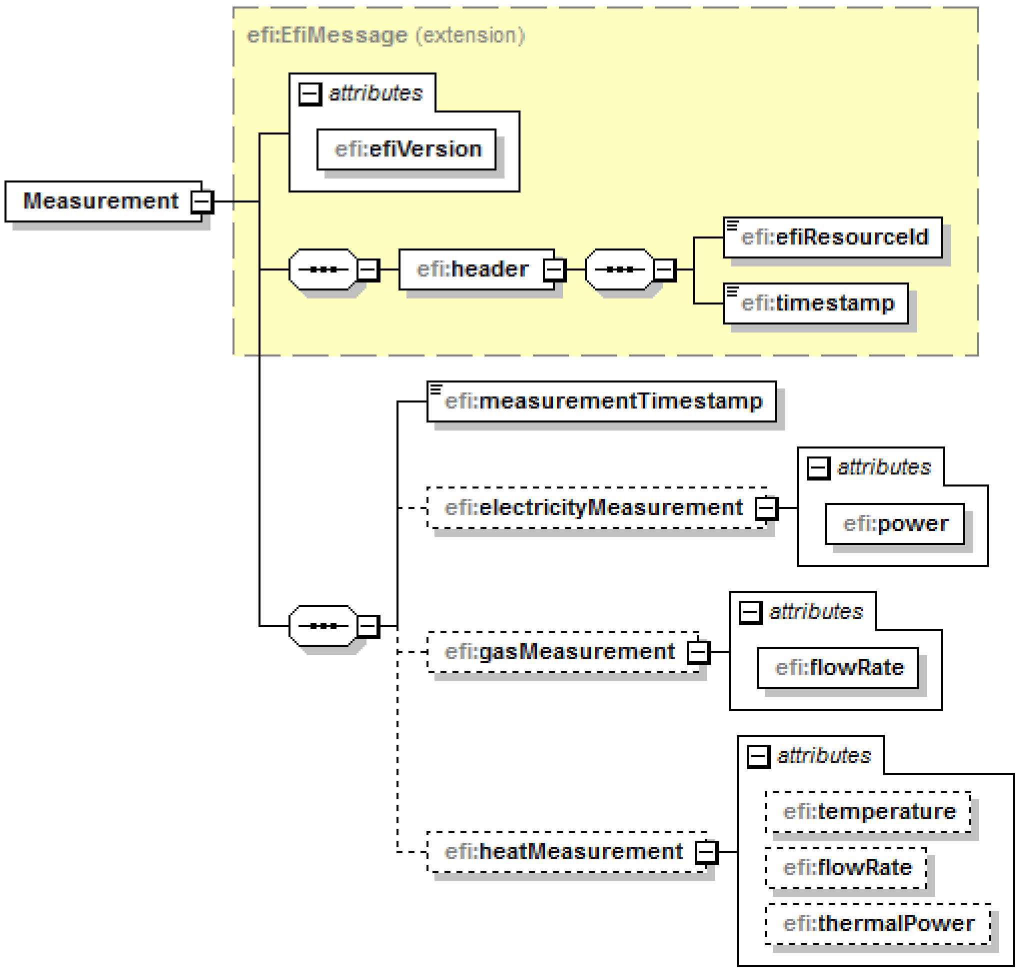

FIGURE 13: OVERVIEW OF THE MEASUREMENT CLASS

The Measurement class is derived from EfiMessage (see section 3.2) and in addition features the following attributes:

Attribute	Description
measurementTimestamp (dateTime)	This attribute indicates the moment in time when the measurement was recorded.
electricityMeasurement	An electricityMeasurement is optional.
power (double)	If the electricityMeasurement is present it must contain a power attribute. The unit for the power measurement is watt.
gasMeasurement	A gasMeasurement is optional.
flowRate (double)	If the gasMeasurement is present it must contain a flowRate attribute. The unit for the flowRate measurement is m^3/s (cubic meters per second).
heatMeasurement	A heatMeasurement is optional. If it is present one or more of the following attributes have to be present as

	well: temperature, flowRate or thermalPower.
temperature (double)	If the heatMeasurement is present this optional attribute may be provided. The unit for the temperature measurement is degrees Celsius.
flowRate (double)	If the heatMeasurement is present this optional attribute may be provided. The unit for the flowRate measurement is m^3/s (cubic meters per second).
thermalPower (double)	If the heatMeasurement is present this optional attribute may be provided. The unit for the thermalPower measurement is watt.

TABLE 6: ATTRIBUTES OF THE MEASUREMENT CLASS

4 Inflexible category

The Inflexible category represents those devices that simply follow a (predetermined) profile without the ability to shift it or to alter the shape of the profile. Examples of such devices are PV panels, wind turbines and brown goods.

Although there is not much flexibility in this category, the CEM still needs to take these devices into account as they are part of the complete energy mix that has to be managed.

The only source of flexibility in the category comes from devices that are capable of curtailment. Think of PV panels for instance that can limit to their output if necessary.

4.1 Typical Inflexible Message Flow

Figure 14 shows the message flow of the Inflexible category. Please note that all messages are derived from the generic messages shown in Figure 4.

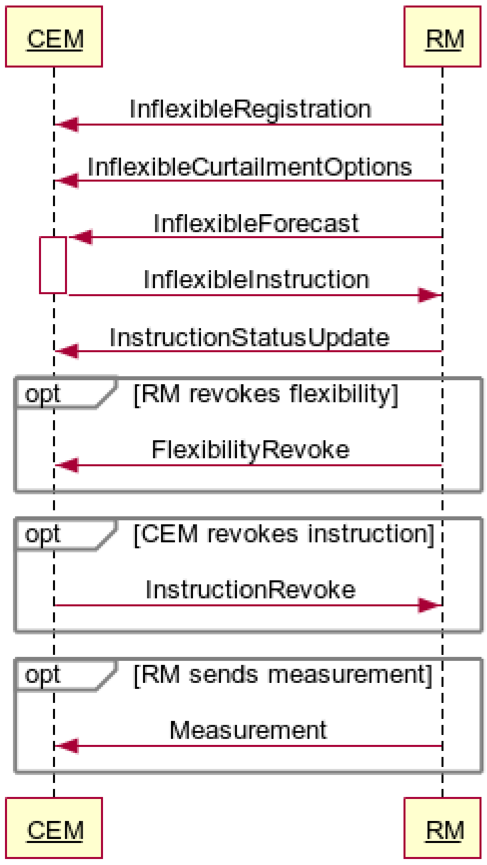

FIGURE 14: A TYPICAL INFLEXIBLE MESSAGE FLOW

In the following paragraphs each message is described in detail, with the exception of the messages that have already been described in previous chapters: InstructionStatusUpdate (see 3.6), FlexibilityRevoke (see 3.7), InstructionRevoke (see 3.8) and Measurement (see 3.9).

4.2 InflexibleRegistration

This message is used by Resource Managers that represent an inflexible device to inform the CEM of the capabilities of that device.

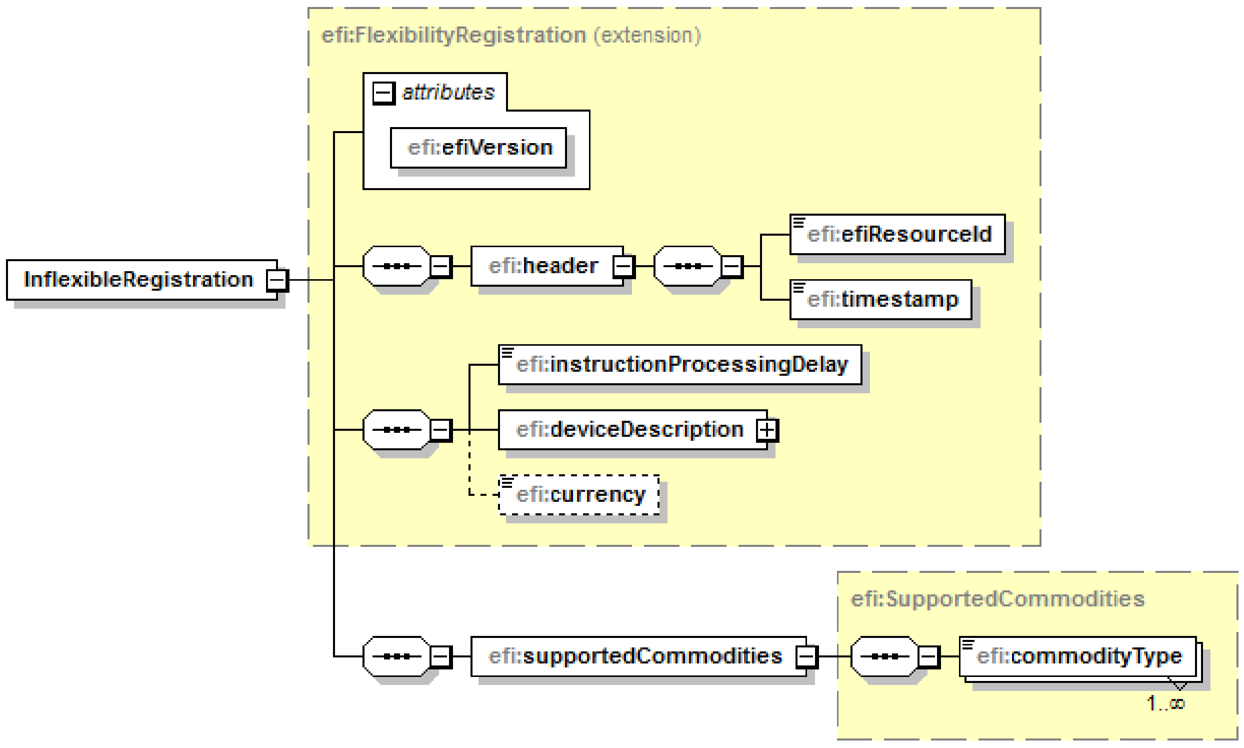

FIGURE 15: OVERVIEW OF THE INFLEXIBLEREGISTRATION CLASS

The InflexibleRegistration class is derived from FlexibilityRegistration (see section 3.3) and in addition features the following attributes:

Attribute	Description
commodityType (CommodityEnum)	The attribute commodityType indicates which commodities are supported by this resource. It is possible to specifiy multiple commodityTypes. The possible values are listed in the CommodityEnum enumeration: • ELECTRICITY • GAS • HEAT

TABLE 7: ATTRIBUTES OF THE INFLEXIBLEREGISTRATION CLASS

4.3 InflexibleUpdate

The InflexibleUpdate is a class that is not being used as a message directly (that is why it is not visible in Figure 14). It serves a parent class from which more specific InflexibleUpdate messages are derived.

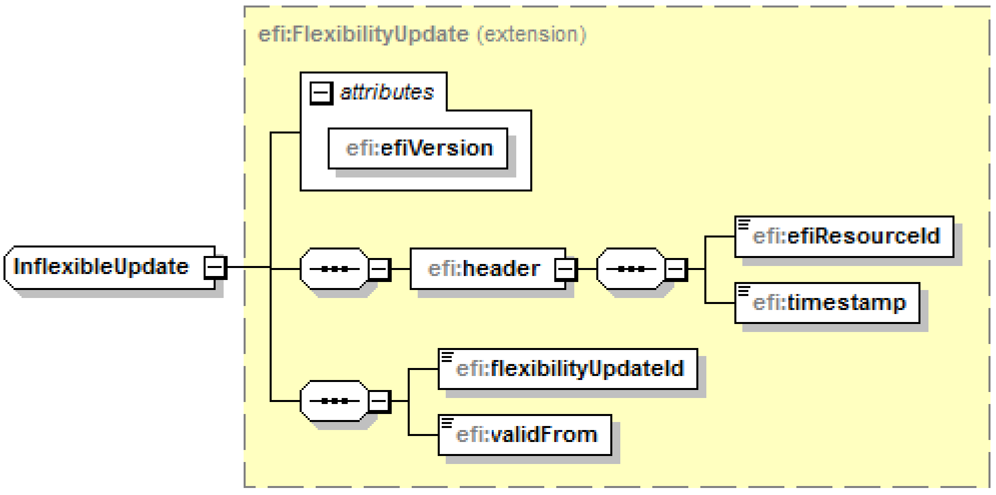

FIGURE 16: OVERVIEW OF THE INFLEXIBLEUPDATE CLASS

The InflexibleUpdate class is derived from FlexibilityUpdate (see section 3.43.2) and does not feature additional attributes.

4.4 InflexibleCurtailmentOptions

If an inflexible device is capable of curtailment a Resource Manager may send this message to the CEM to inform it about the curtailment capabilities.

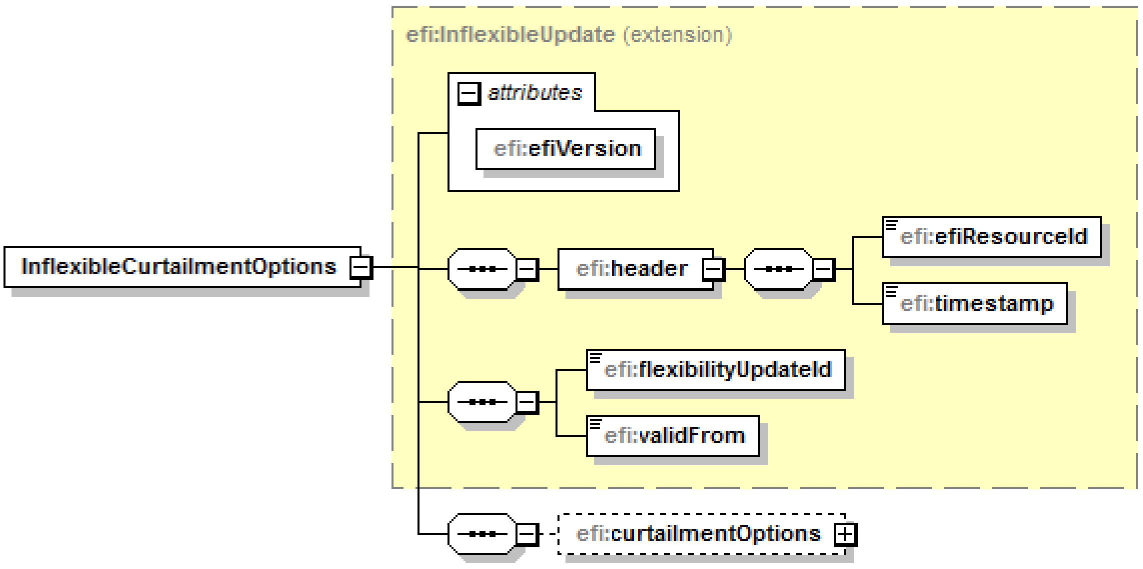

FIGURE 17: OVERVIEW OF THE INFEXIBLECURTAILMENTOPTIONS CLASS

The InflexibleCurtailmentOptions class is derived from InflexibleUpdate (see section 4.3 3.2) and in addition features the following attributes:

Attribute	Description

curtailmentOptions (CurtailmentOptions)	This attribute describes the possible curtailmentOptions and is of the type CurtailmentOptions. This type is explained in paragraph 4.4.1.

4.4.1 CurtailmentOptions

The CurtailmentOptions class describes possible curtailment options for an inflexible device.

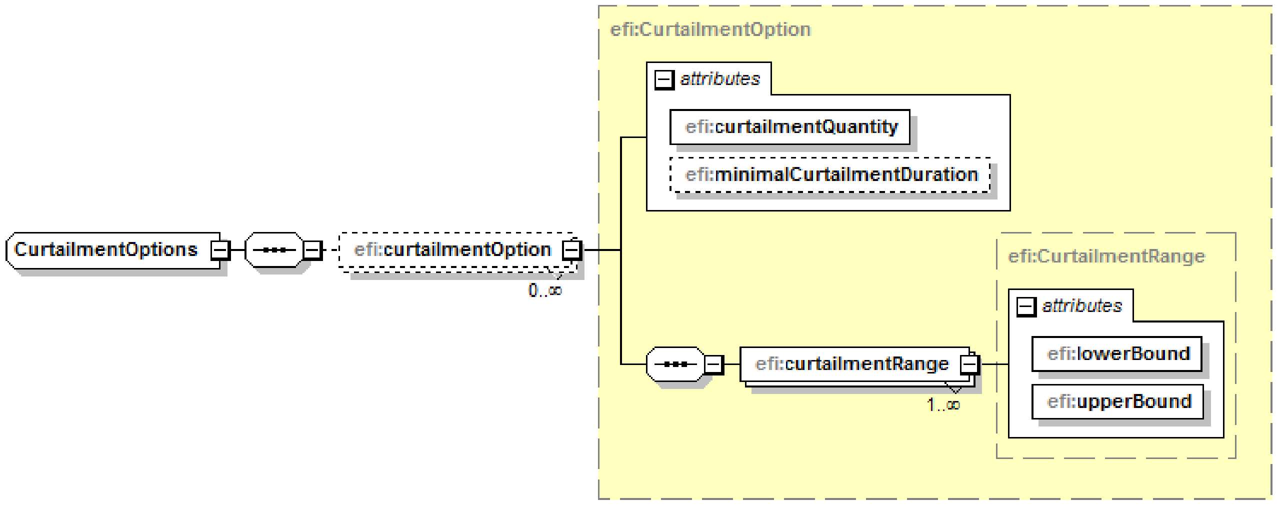

FIGURE 18: OVERVIEW OF THE CURTAILMENTOPTIONS CLASS

The CurtailmentOptions class has the following attributes:

Attribute	Description
curtailmentOption (CurtailmentOption)	This attribute describes a single curtailmentOption. There can be multiple curtailmentOption attributes within the CurtailmentOptions class. The CurtailmentOption type is explained in paragraph 4.4.2.

TABLE 8: ATTRIBUTES OF THE CURTAILMENTOPTIONS CLASS

4.4.2 CurtailmentOption

This class describes a single CurtailmentOption and has the following attributes:

Attribute	Description
curtailmentQuantity (CurtailmentQuantity)	This attribute indicates which quantity may be curtailed with this CurtailmentOption. It is of the type CurtailmentQuantity, which is an enumeration that is further explained in paragraph 4.4.3.
minimalCurtailmentDuration (duration)	It could be that a curtailment option is only available if it is used for a minimal period of time. If that is the case it

	can be indicated with this optional attribute.
curtailmentRange (CurtailmentRange)	This attribute is of the CurtailmentRange type. There can be multiple curtailmentRange attributes within the CurtailmentOption class. This type is explained in paragraph 4.4.4.

TABLE 9: ATTRIBUTES OF THE CURTAILMENTOPTION CLASS

4.4.3 CurtailmentQuantity

This enumeration specifies which quantities can be curtailed. The following values are valid:

- ELECTRICITY.POWER
- GAS.FLOWRATE
- HEAT.TEMPERATURE
- HEAT.FLOWRATE
- HEAT.THERMALPOWER

4.4.4 CurtailmentRange

This class describes a single curtailment range or value. If the lowerbound and upperboud are equal it represents a single value, otherwise a continuous range is described. It has the following attributes:

Attribute	Description
lowerbound (double)	The lower bound of the range or the value.
upperbound (double)	The upper bound of the range or the value.

TABLE 10: ATTRIBUTES OF THE CURTAILMENTRANGE CLASS

4.5 InflexibleForecast

Although the Inflexibible category does not provide any form of flexibility with the exception of curtailment, it may provide a forecast so that the CEM can take this into account in its matching process. for The InflexibleForecast message contains a consumption and/or production forecast for an inflexible device.

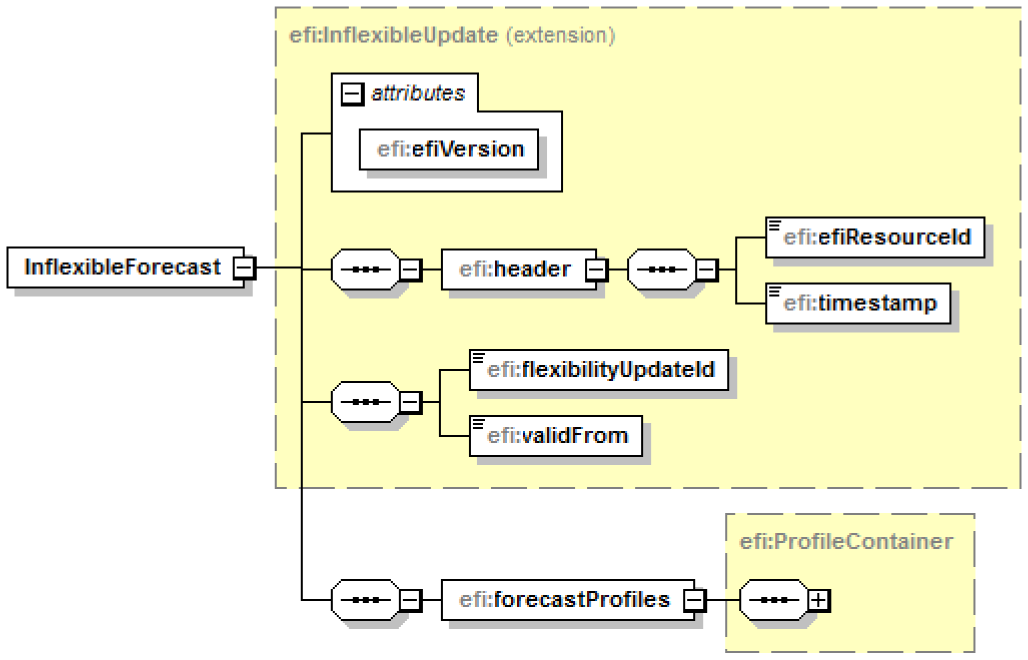

FIGURE 19: OVERVIEW OF THE INFLEXIBLEFORECAST CLASS

The InflexibleForecast class is derived from InflexibleUpdate (see section 4.3 3.2) and in addition features the following attributes:

Attribute	Description
forecastProfiles (ProfileContainer)	The forecastProfiles attribute is of the type ProfileContainer and may contain one or more profiles for electricity, gas or heat. The ProfileContainer type is explained further in paragraph 4.5.1.

TABLE 11: ATTRIBUTES OF THE INFLEXIBILEFORECAST CLASS

4.5.1 ProfileContainer

The ProfileContainer class can contain an electricity -, gas - or heat profile or any combination of these three.

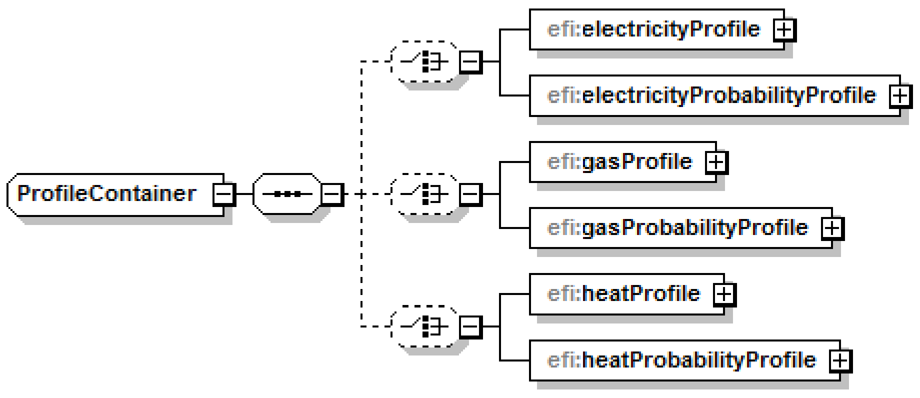
ProfileContainer
efi:electricityProfile
efi:electricityProbabilityProfile
efi:gasProfile
efi:gasProbabilityProfile
efi:heatProfile
efi:heatProbabilityProfile

FIGURE 20: OVERVIEW OF THE PROFILECONTAINER CLASS

As can be seen from Figure 20 there are two versions of each profile (electricity, gas and heat) available: a regular one and a profile containing probability information. If a forecast for a specific commodity is provided a choice has to be made between the normal profile and the probability profile. The following table describes the attributes of the ProfileContainer class:

Attribute	Description
electricityProfile (ElectricityProfile)	This optinal attribute is of the type ElectricityProfile and is further explained in paragraph 4.5.2.
electricityProbabilityProfile (ElectricityProbabilityProfile)	This optional attribute is of the type ElectricityProbabilityProfile and is further explained in paragraph 4.5.3.
gasProfile (GasProfile)	This optional attribute is of the type GasProfile and is further explained in paragraph 4.5.6.
gasProbabilityProfile (GasProbabilityProfile)	This optional attribute is of the type GasProbabilityProfile and is further explained in paragraph 4.5.7.
heatProfile (HeatProfile)	This optional attribute is of the type HeatProfile and is further explained in paragraph 4.5.8.
heatProbabilityProfile (HeatProbabilityProfile)	This optional attribute is of the type HeatProbabilityProfile and is further explained in paragraph 4.5.9.

TABLE 12: ATTRIBUTES OF THE PROFILECONTAINER CLASS

4.5.2 ElectricityProfile

The ElectrcityProfile expresses a power profile over time. The timeslots can have a variable size in order to make the resolution as fine grained as needed. Please note that the ElectricityProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

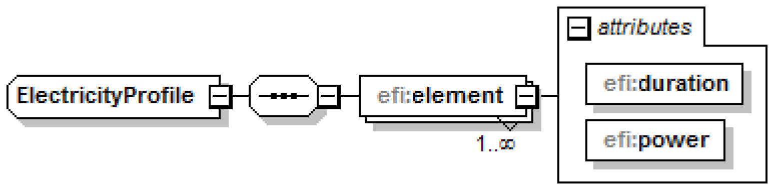

FIGURE 21: OVERVIEW OF THE ELECTRICITYPROFILE CLASS

The ElectricityProfile class consists of 1 or more elements. Each element represents a timeslot and has the following attributes:

Attribute	Description

duration (duration)	This is the duration of the element. The shortest possible duration is 1 second.
power (double)	This is the power value of this element. The unit of power in an ElectricityProfile is watt.

TABLE 13: ATTRIBUTES OF THE ELECTRICITYPROFILE CLASS

4.5.3 ElectricityProbabilityProfile

Like the ElectricityProfile the ElectricityProbabilityProfile class expresses a power profile over time. The difference is that this class also provides probability information about the expected power value in the profile elements. Please note that the ElectricityProbabilityProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

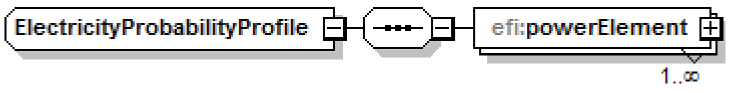

FIGURE 22: OVERVIEW OF THE ELECTRICITYPROBABILITYPROFILE CLASS

The ElectricityProbabilityProfile class has the following attributes:

Attribute	Description
powerElement (ProbabilityAttributesWithDuration)	Within an ElectricityProbabilityProfile there are one or more powerElement attributes. This attribute is of the type ProbabilityAttributesWithDuration and is further explained in paragraph 4.5.4. Within the context of this ElectricityProbabilityProfile class all doubles in the ProbabilityAttributesWithDuration class must be interpreted as having the watt unit.

TABLE 14: ATTRIBUTES OF THE ELECTRICITYPROBABILITYPROFILE CLASS

4.5.4 ProbabilityAttributesWithDuration

The ProbabilityAttributesWithDuration class defines a timeslot with a duration and an expected value together with a probability distribution of that expected value.

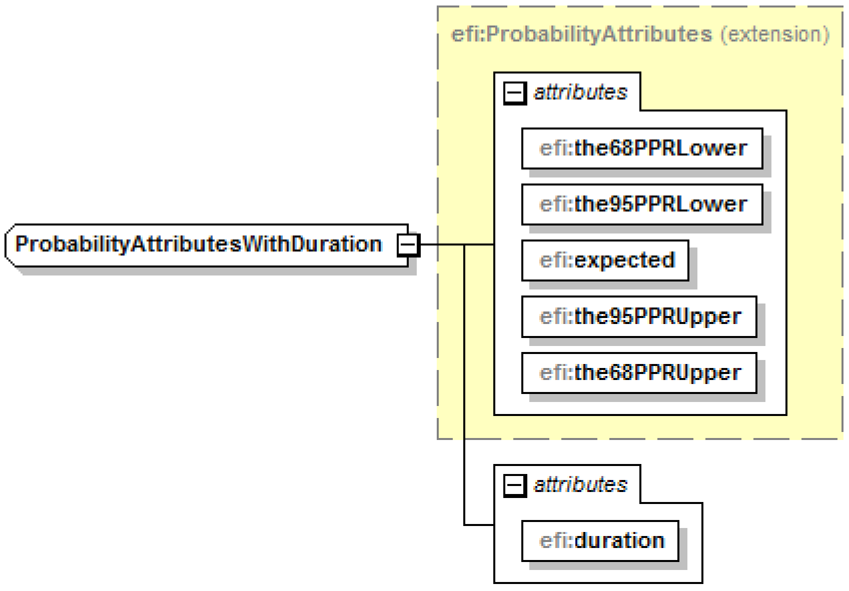

FIGURE 23: OVERVIEW OF THE PROBABILITYATTRIBUTESWITHDURATION CLASS

The ProbabilityAttributesWithDuration class is derived from ProbabilityAttributes (see section 4.5.5) and in addition features the following attributes:

Attribute	Description
duration (duration)	This is the duration of the element. The shortest possible duration is 1 second.

TABLE 15: ATTRIBUTES OF THE PROBABILITYATTRIBUTESWITHDURATION CLASS

4.5.5 ProbabilityAttributes

This class is being used to express uncertainty about an expected value. In most cases one would choose a normal or Gaussian distribution and specify the mean (μ) and the standard deviation (σ), whereby 68% of the range would fall within μ ± σ and 95% within μ ± 2σ. However a Gaussian distribution is not always the correct one to use, because a distribution may also be asymmetric. Consider a PV panel on a cloudy day for instance. It will be most likely that the PV panel will not produce any or only very little energy. The chance that a significant amount of energy will be produced is very small. Therefore the centre of gravity of this distribution will be around zero, yet it will not extend below zero, since the PV panel will never consume energy.

The PV panel example underlines the need to support different distributions. One way of doing this would be to explicitly specify the distribution type that is being used. However that implies that both sides need to be aware of the full set of possible distribution types.

Another way of handling this problem is to not specify a particular distribution at all. Instead ProbabilityAttributes class provides the expected value as well as the ranges in which 68 and 95 percent of the values fall. For both ranges a lower and upper bound is given. This flexible solution also allows the expression of asymmetric distributions as described in the PV panel example.

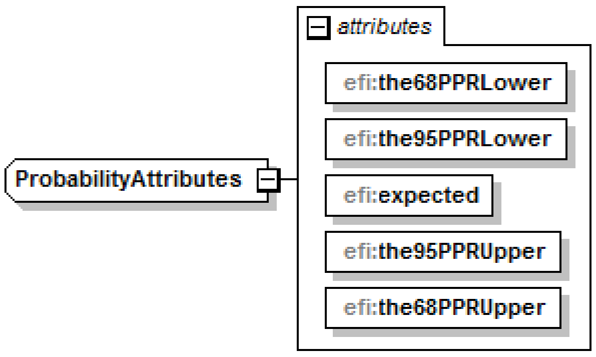

FIGURE 24: OVERVIEW OF THE PROBABILITYATTRIBUTES CLASS

This class has the following attributes:

Attribute	Description
expected (double)	The expected value.
the68pprLowerBound (double)	The lower bound of the 68 percent range.
the68pprUpperBound (double)	The upper bound of the 68 percent range.
the95pprLowerBound (double)	The lower bound of the 95 percent range.
the95pprUpperBound (double)	The upper bound of the 95 percent range.

TABLE 16: ATTRIBUTES OF THE PROBABILITYATTRIBUTES CLASS

4.5.6 GasProfile

The GasProfile expresses a flowRate profile over time. The timeslots can have a variable size in order to make the resolution as fine grained as needed. Please note that the GasProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

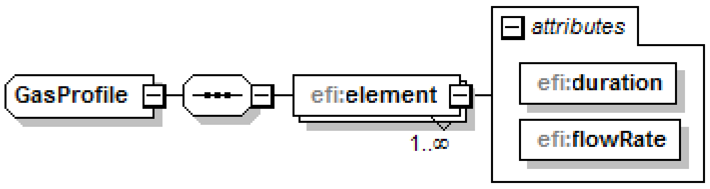

FIGURE 25: OVERVIEW OF THE GASPROFILE CLASS

The GasProfile class consists of 1 or more elements. Each element represents a timeslot and has the following attributes:

Attribute	Description
duration (duration)	This is the duration of the element. The shortest possible duration is 1 second.
flowRate (double)	This is the flow rate value of this element. The unit of power in an GasProfile is m^3/s.

TABLE 17: ATTRIBUTES OF THE GASPROFILE CLASS

4.5.7 GasProbabilityProfile

Like the GasProfile the GasProbabilityProfile class expresses a flow rate profile over time. The difference is that this class also provides probability information about the expected flow rate value in the profile elements. Please note that the GasProbabilityProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

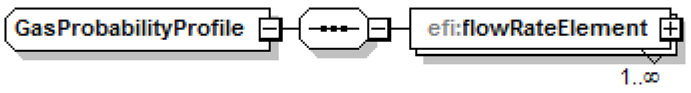

FIGURE 26: OVERVIEW OF THE GASPROBABILITYPROFILE CLASS

The GasProbabilityProfile class has the following attributes:

Attribute	Description
flowRateElement (ProbabilityAttributesWithDuration)	Within a GasProbabilityProfile there are one or more flowRateElement attributes. This attribute is of the type ProbabilityAttributesWithDuration and is further explained in paragraph 4.5.4. Within the context of this GasProbabilityProfile class all doubles in the ProbabilityAttributesWithDuration class must be interpreted as having the m^3/s unit.

FIGURE 27: ATTRIBUTES OF THE GASPROBABILITYPROFILE CLASS

4.5.8 HeatProfile

The HeatProfile expresses a heat profile over time. The timeslots can have a variable size in order to make the resolution as fine grained as needed. Please note that the HeatProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

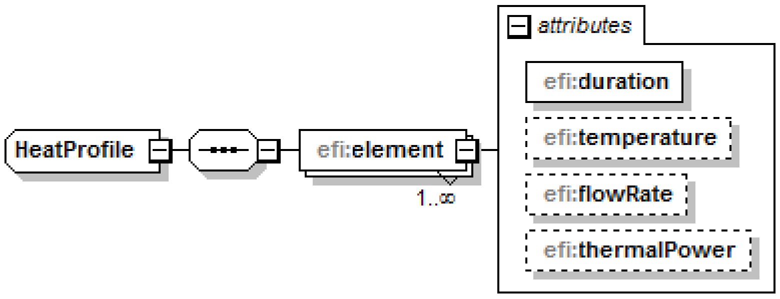

FIGURE 28: OVERVIEW OF THE HEATPROFILE CLASS

The HeatProfile class consists of 1 or more elements. Each element represents a timeslot with a duration and one or more of the following values: temperature, flowRate or thermalPower. The attributes are described in more detail below:

Attribute	Description
duration (duration)	This is the duration of the element. The shortest possible duration is 1 second.
temperature (double)	This optional attribute is the temperature value of this element. The unit of temperature in a HeatProfile is degrees celsius.
flowrate (double)	This optional attribute is the flow rate value of this element. The unit of flow rate in a HeatProfile is m^3/s.
thermalPower (double)	This optional attribute is the thermal power value of this element. The unit of thermal power in a HeatProfile is watt.

FIGURE 29: ATTRIBUTES OF THE HEATPROFILE CLASS

4.5.9 HeatProbabilityProfile

Like the HeatProfile the HeatProbabilityProfile class expresses a heat profile over time. The difference is that this class also provides probability information about the expected values in the profile elements. Please note that the HeatProbabilityProfile itself does not provide a start time. The start time of the profile is equal to the validFrom timestamp of the InflexibleForecast message (see 4.5).

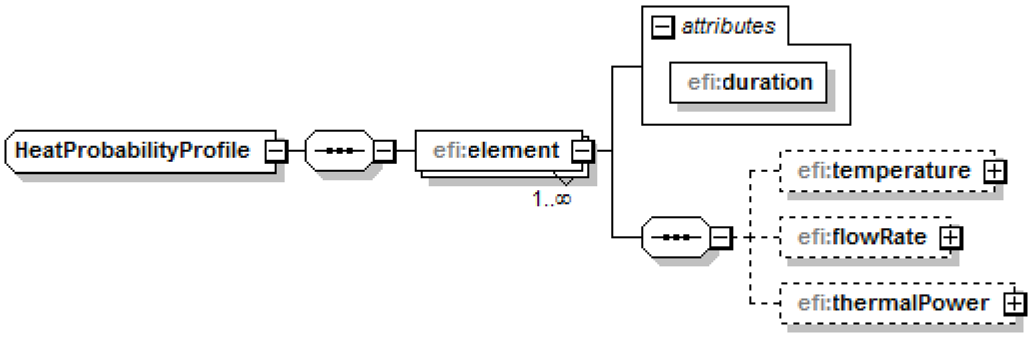

FIGURE 30: OVERVIEW OF THE HEATPROBABILITYPROFILE CLASS

The HeatProabilityProfile class consists of 1 or more elements. Each element represents a timeslot with a duration and one or more of the following probability values: temperature, flowrate or thermalPower. The attributes are described in more detail below:

Attribute	Description
duration (duration)	This is the duration of the element. The shortest possible duration is 1 second.
temperature (ProbabilityAttributes)	This optional attribute expresses the expected temperature value of this element. The type of the attribute is ProbabilityAttributes and is further explained in paragraph 4.5.5. Within the context of this temperature attribute all doubles in the ProbabilityAttributes class must be interpreted as having the degrees Celsius unit.
flowRate (ProbabilityAttributes)	This optional attribute expresses the expected flowRate value of this element. The type of the attribute is ProbabilityAttributes and is further explained in paragraph 4.5.5. Within the context of this flowRate attribute all doubles in the ProbabilityAttributes class must be interpreted as having the m^3/s unit.
thermalPower (ProbabilityAttributes)	This optional attribute expresses the expected thermalPower value of this element. The type of the attribute is ProbabilityAttributes and is further explained in paragraph 4.5.5. Within the context of this thermalPower attribute all doubles in the ProbabilityAttributes class must be interpreted as having the watt unit.

4.6 InflexibleInstruction

This instruction message is being sent from the CEM to the RM in order to realize a desired curtailment profile over time. This means that the consumption or production of the device that is managed by the RM will be cut off at the level(s) specified in the curtailment profile.

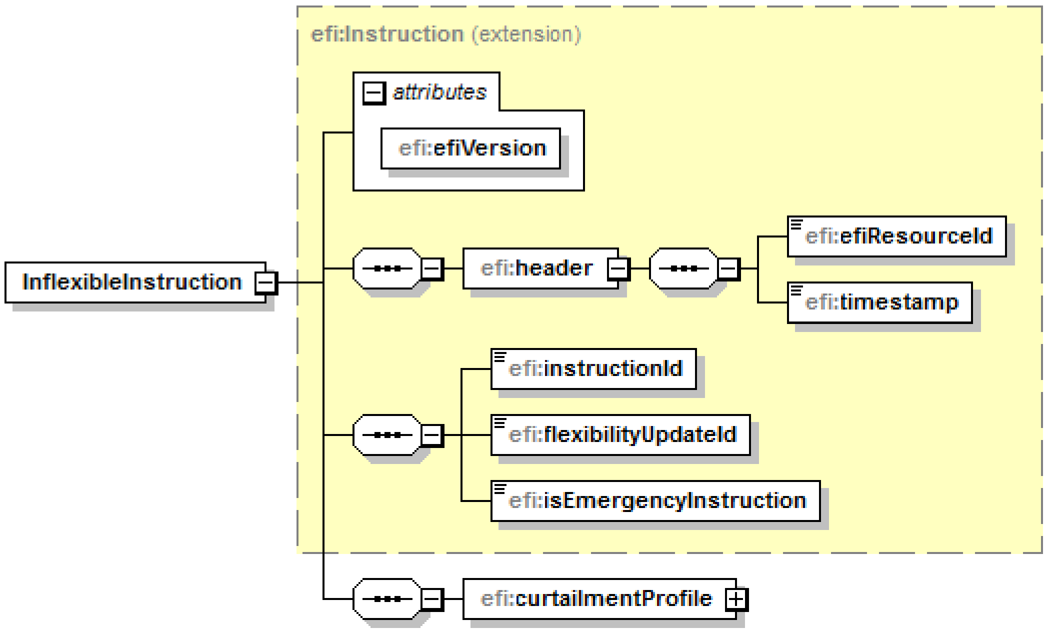

FIGURE 31: OVERVIEW OF THE INFLEXIBLEINSTRUCTION CLASS

The InflexibleInstruction class is derived from Instruction (see section 3.53.2) and in addition features the following attributes:

Attribute	Description
curtailmentProfile (ProfileContainer)	The curtailmentProfile attribute is of the type CurtailmentProfile and contains a profile that specifies how this inflexibile device should be curtailed over time. The CurtailmentProfile type is explained further in paragraph 4.5.1.

TABLE 18: ATTRIBUTES OF THE INFLEXIBILEINSTRUCTION CLASS

4.6.1 CurtailmentProfile

This class contains a profile which specifies over time to which levels the production or consumption of the inflexible device needs to be curtailed.

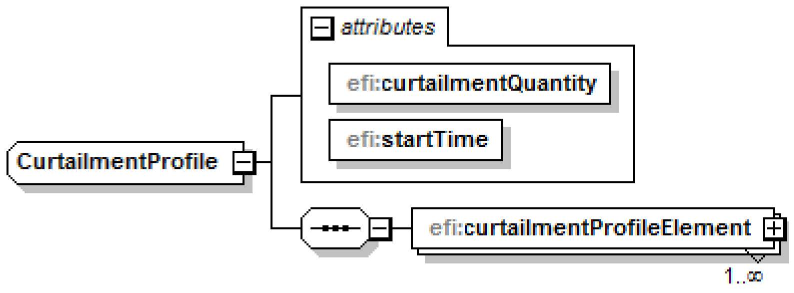

FIGURE 32: OVERVIEW OF THE CURTAILMENTPROFILE CLASS

The CurtailmentProfile class has the following attributes:

Attribute	Description
curtailmentQuantity (CurtailmentQuantity)	This attribute specifies which quantity is being curtailed in this CurtailmentProfile. It is enumeration of the type CurtailmentQuantity and is further explained in paragraph 4.4.3.
startTime (dateTime)	This attribute of the type dateTime indicates the starting time of this curtailmentProfile.
curtailmentProfileElement (CurtailmentProfileElement)	This attribute may occur one or more times. It represents a timeslot of this CurtailmentProfile. The type of this attribute is CurtailmentProfileElement and is further explained in paragraph 4.6.2.

TABLE 19: ATTRIBUTES OF THE CURTAILMENTPROFILE CLASS

4.6.2 CurtailmentProfileElement

The CurtailmentProfileElement class represent a single timeslot in a curtailment profile.

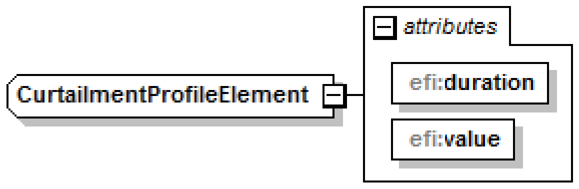

FIGURE 33: OVERVIEW OF THE CURTAILMENTPROFILEELEMENT CLASS

The CurtailmentProfileElement class has the following attributes:

Attribute	Description
duration (dateTime)	This is the duration of the element. The shortest possible

	duration is 1 second.
value (double)	This is the curtailment value for the timeslot that this CurtailmentProfileElement class represents. The production or consumption of the inflexible device should remain between zero and this value. That means that in the case of a consumption curtailment (positive value) the consumption may never be greater than this value and that in the case of a production curtailment (negative value) the production may never be smaller.

TABLE 20: ATTRIBUTES OF THE CURTAILMENTPROFILEELEMENT CLASS

4.7 InstructionStatusUpdate

See paragraph 3.6 for a description of this message.

4.8 FlexibilityRevoke

See paragraph 3.7 for a description of this message.

4.9 InstructionRevoke

See paragraph 3.8 for a description of this message.

4.10 Measurement

See paragraph 3.9 for a description of this message.

5 Shiftable category

A Shiftable appliance is an appliance which needs to run a predetermined program within a certain time window. The start time of the program can be scheduled by an energy service with the only requirement that it has to be finished by a given deadline. White goods are typical examples of appliances which can adopt this Shiftable behaviour.

5.1 Typical Inflexible Message Flow

Figure 34 shows the message flow of the Shiftable category. Please note that all messages are derived from the generic messages shown in Figure 4.

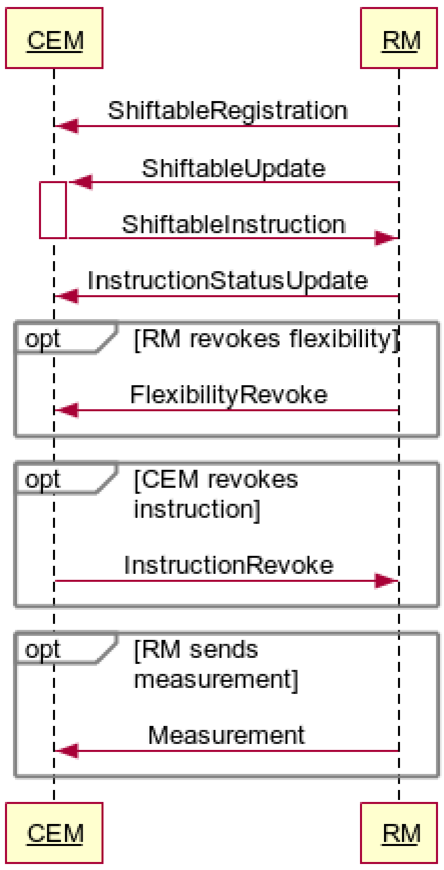

FIGURE 34: A TYPICAL SHIFTABLE MESSAGE FLOW

In the following paragraphs each message is described in detail, with the exception of the messages that have already been described in previous chapters: InstructionStatusUpdate (see 3.6), FlexibilityRevoke (see 3.7), InstructionRevoke (see 3.8) and Measurement (see 3.9).

5.2 ShiftableRegistration

This message is used by Resource Managers that represent a shiftable device to inform the CEM of the capabilities of that device.

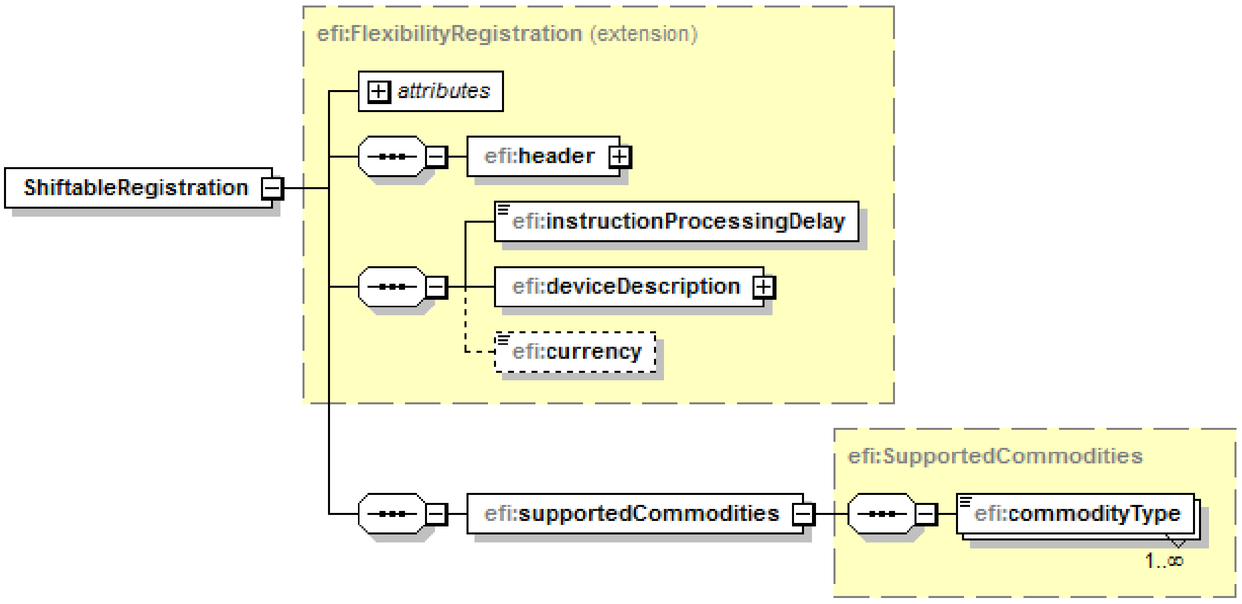

FIGURE 35: OVERVIEW OF THE SHIFTABLEREGISTRATION CLASS

The ShiftableRegistration class is derived from the FlexibilityRegistration (see section 3.3) and in addition features the following attributes:

Attribute	Description
commodityType (CommodityEnum)	The attribute commodityType indicates which commodities are supported by this resource. It is possible to specify multiple commodityTypes. This attribute is of the type CommodityEnum which is further explained in paragraph 5.2.1.

TABLE 21: ATTRIBUTES OF THE SHIFTABLEREGISTRATION CLASS

5.2.1 CommodityEnum

This enumeration specifies which commodities may be supported by a resource. The following values are valid:

- ELECTRICITY
- GAS
- HEAT

5.3 ShiftableUpdate

This message is sent by the Resource Manager to inform the CEM of the time shift options of the device.

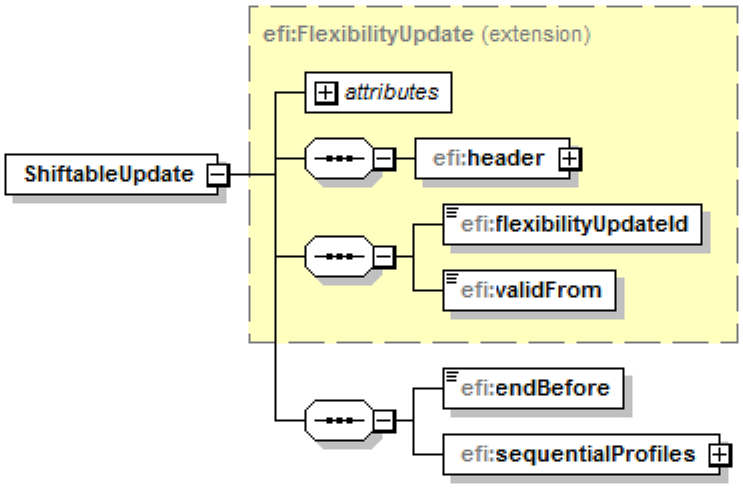

FIGURE 36: OVERVIEW OF THE SHIFTABLEUPDATE CLASS

The ShiftableUpdate class is derived from the FlexibilityUpdate class (see 3.4) and in addition features the following attributes:

Attribute	Description
endBefore (dateTime)	The moment in time where every sequential profile in this ShiftableUpdate must be finished.
sequentialProfiles (SequentialProfiles)	This attribute describes the sequential profiles for this ShiftableUpdate and is of the type SequentialProfiles. This type is explained in paragraph 5.3.1.

TABLE 22: ATTRIBUTES OF THE SHIFTABLEUPDATE CLASS

5.3.1 SequentialProfiles

A complete profile for a Shiftable device may actually consist of multiple sequential profiles separated by a variable pause. An example use case for sequential profiles would be a washing machine that has the ability to separate its program into a separate phases (heating, tumble drying, etc.). Between each phase a pause of variable length can be inserted. This provides even more flexibility than just shifting the start time of the complete program.

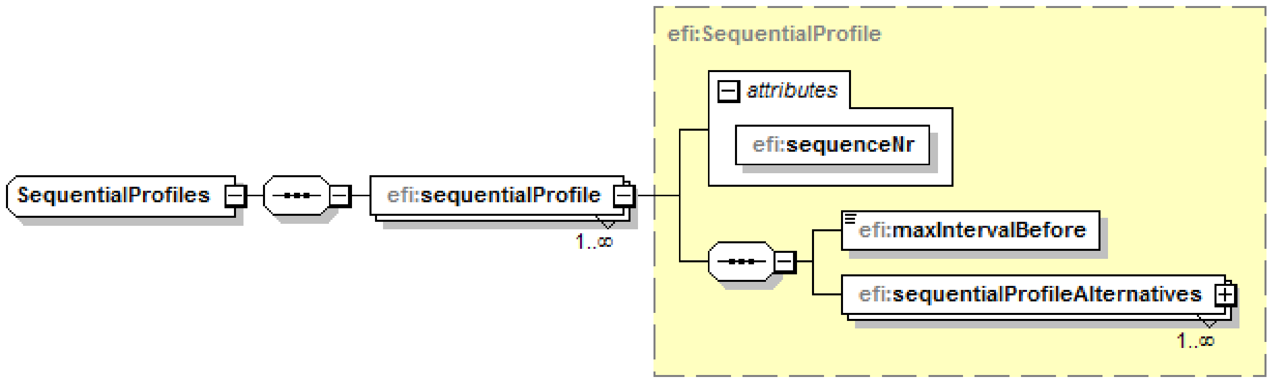

FIGURE 37: OVERVIEW OF THE SEQUENTIALPROFILES CLASS

The SequentialProfiles class has the following attributes:

Attribute	Description
sequentialProfile (SequentialProfile)	This attribute describes a single sequentialProfile. There can be multiple sequentialProfile attributes within the SequentialProfiles class. The SequentialProfile type is explained in paragraph 5.3.2.

TABLE 23: ATTRIBUTES OF THE SEQUENTIALPROFILES CLASS

5.3.2 SequentialProfile

This class describes a single SequentialProfile and has the following attributes:

Attribute	Description
sequenceNr (int)	This is a unique ID to identify this SequentialProfile.
maxIntervalBefore (duration)	The maximum time between the end of the last SequentialProfile and the start of this SequentialProfile. The maxIntervalBefore of the first sequential profile is typically 0 seconds. This attribute is of the type duration.
sequentialProfileAlternatives (SequentialProfileAlternatives)	Within a SequentialProfile there may be multiple alternative profiles to achieve the same goal. These are expressed with the sequentialProfileAlternatives attribute which is of the type SequentialProfileAlternatives and is further explained in paragraph .

TABLE 24: ATTRIBUTES OF THE SEQUENTIALPROFILE CLASS

5.3.3 SequentialProfileAlternatives

The SequentialProfileAlternatives class may be used to provide multiple alternative profiles for achieving the same goal. Consider a heating phase of washing machine. In order to reach the desired temperature, one could opt for a high power profile that needs less time to heat the water or a low power alternative that takes longer to reach the same temperature.

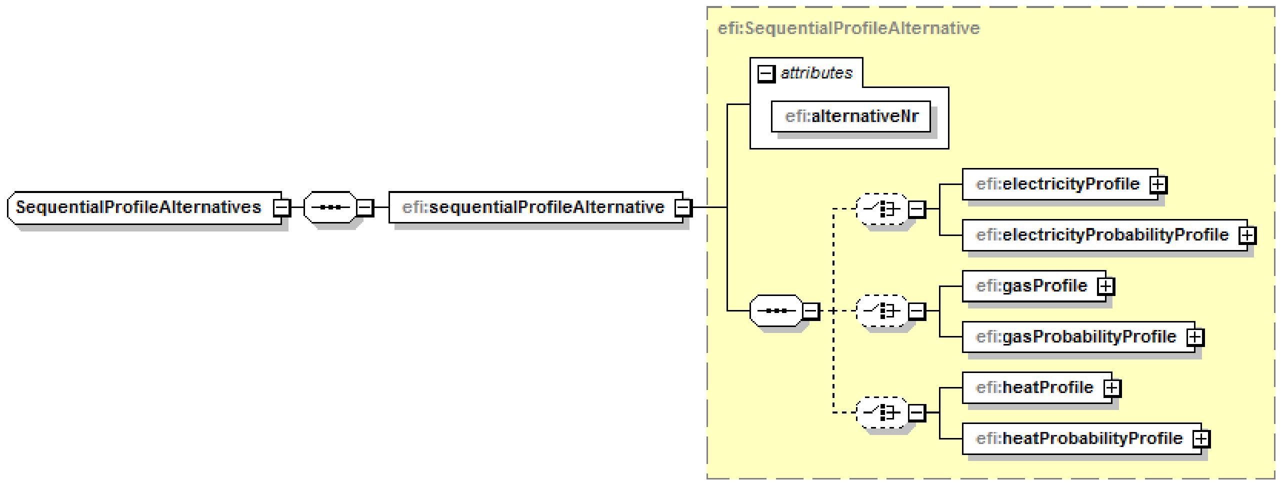
efi:SequentialProfileAlternative
attributes
efi:alternativeNr
SequentialProfileAlternatives
efi:sequentialProfileAlternative
efi:electricityProfile
efi:electricityProbabilityProfile
efi:gasProfile
efi:gasProbabilityProfile
efi:heatProfile
efi:heatProbabilityProfile

FIGURE 38: OVERVIEW OF THE SEQUENTIALPROFILEALTERNATIVES CLASS

The SequentialProfileAlternatives class features the following attributes:

Attribute	Description
sequentialProfileAlternative (SequentialProfileAlternative)	This attribute describes a single sequentialProfileAlternative. There can be multiple sequentialProfileAlternative attributes within the SequentialProfileAlternatives class. The SequentialProfileAlternative type is explained in paragraph 5.3.4.

TABLE 25: ATTRIBUTES OF THE SEQUENTIALPROFILEALTERNATIVES CLASS

5.3.4 SequentialProfileAlternative

The SequentialProfileAlternative class describes a single alternative for a SequentialProfile. It is an extension of the ProfileContainer class (see 4.5.1) and in addition has the following attributes:

Attribute	Description
alternativeNr(int)	This is a unique ID to identify this SequentialProfileAlternative.

TABLE 26: ATTRIBUTES OF THE SEQUENTIALPROFILEALTERNATIVE CLASS

5.4 ShiftableInstruction

This instruction message is being sent from the CEM to the RM. It contains the selection of particular sequential profile alternatives as well as the desired start times for each selected profile.

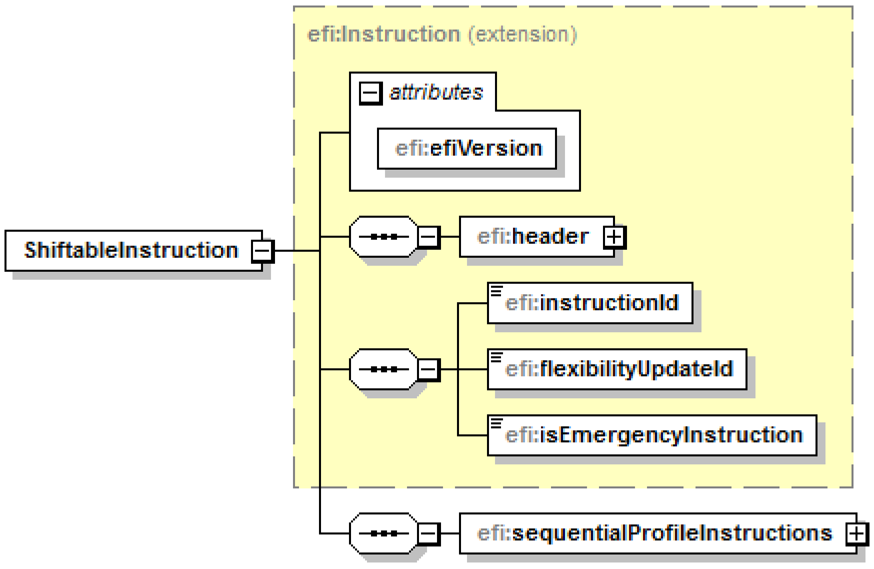

FIGURE 39: OVERVIEW OF THE SHIFTABLEINSTRUCTION CLASS

The ShiftableInstruction class is derived from Instruction (see section 3.5) and in addition features the following attributes:

Attribute	Description
sequentialProfileInstructions (SequentialProfileInstructions)	This attribute contains instructions for each sequential profile. It is of the type SequentialProfileInstructions and is further explained in paragraph 5.4.1.

TABLE 27: ATTRIBUTES OF THE SHIFTABLEINSTRUCTION CLASS

5.4.1 SequentialProfileInstructions

The SequentialProfileInstructions class contains the instruction details for each individual profile.

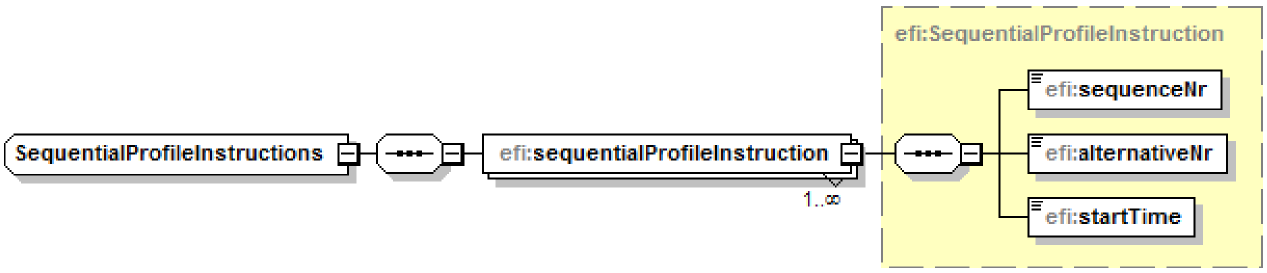
SequentialProfileInstructions
efi:sequentialProfileInstruction
1..∞
efi:SequentialProfileInstruction
efi:sequenceNr
efi:alternativeNr
efi:startTime

FIGURE 40: OVERVIEW OF THE SEQUENTIALPROFILEINSTRUCTIONS CLASS

The SequentialProfiles class features the following attributes:

Attribute	Description
sequentialProfileInstruction (SequentialProfileInstruction)	This attribute describes a single sequentialProfileInstruction. There can be multiple sequentialProfileInstruction attributes within the SequentialProfileInstruction class. The SequentialProfileInstruction type is explained in paragraph 5.4.2.

TABLE 28: ATTRIBUTES OF THE SEQUENTIALPROFILEINSTRUCTIONS CLASS

5.4.2 SequentialProfileInstruction

The SequentialProfileInstruction has the following attributes:

Attribute	Description
sequenceNr (int)	This sequenceNr selects the SequentialProfile that was present in the ShiftableUpdate message that this ShiftableInstruction message is responding to.
alternativeNr (int)	This alternativeNr selects one of the alternative profiles that was present in the SequentialProfile.
startTime (dateTime)	This is the startTime that is requested by the CEM for the alternative profile that is selected in this SequentialProfileInstruction.

6 Adjustable category

The adjustable category can be used for those devices that can easily change from one state to another, without being restricted by a buffer.

- Diesel generator
- Dimmable lights
- Heat pump connected to heat grid

6.1 Typical Adjustable Message Flow

Figure 41 shows the message flow of the Adjustable category. Please note that all messages are derived from the generic messages shown in Figure 4.

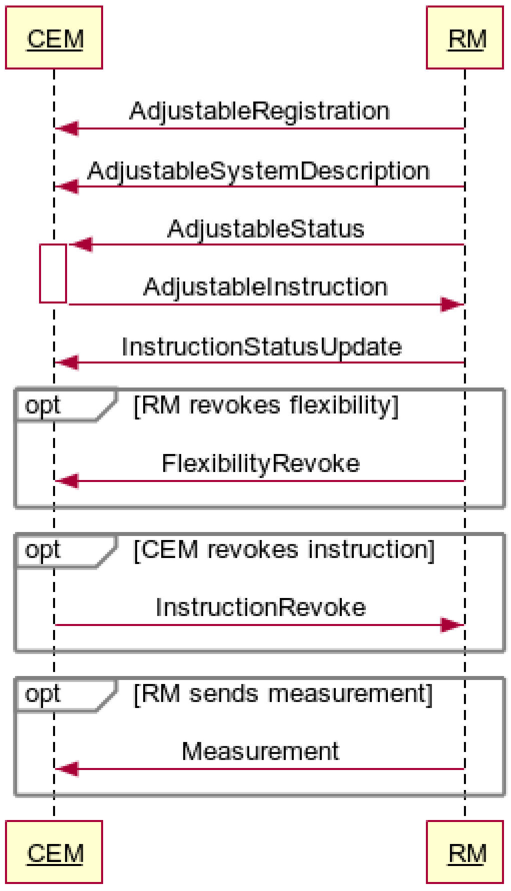

FIGURE 41: A TYPICAL ADJUSTABLE MESSAGE FLOW

In the following paragraphs each message is described in detail, with the exception of the messages that have already been described in previous chapters: InstructionStatusUpdate (see 3.6), FlexibilityRevoke (see 3.7), InstructionRevoke (see 3.8) and Measurement (see 3.9).

6.2 AdjustableRegistration

This message is used by Resource Managers that represent an adjustable device to inform the CEM of the capabilities of that device.

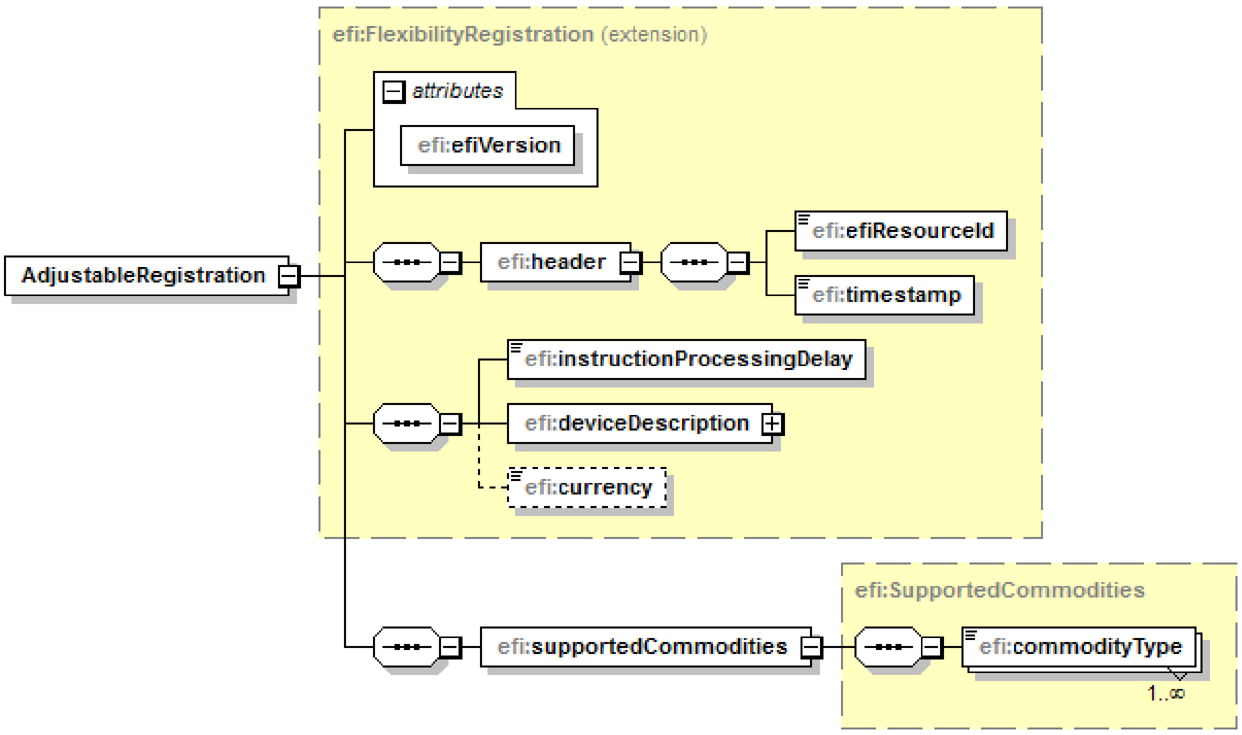

FIGURE 42: OVERVIEW OF THE ADJUSTABLEREGISTRATION CLASS

The AdjustableRegistration class is derived from the FlexibilityRegistration (see section 3.3) and in addition features the following attributes:

Attribute	Description
commodityType (CommodityEnum)	The attribute commodityType indicates which commodities are supported by this resource. It is possible to specify multiple commodityTypes. This attribute is of the type CommodityEnum which is further explained in paragraph 5.2.1.

TABLE 29: ATTRIBUTES OF THE ADJUSTABLEREGISTRATION CLASS

6.3 AdjustableUpdate

There are multiple update messages within the Adjustable class, they are all derived from the AdjustableUpdate class which is shown in Figure 43.

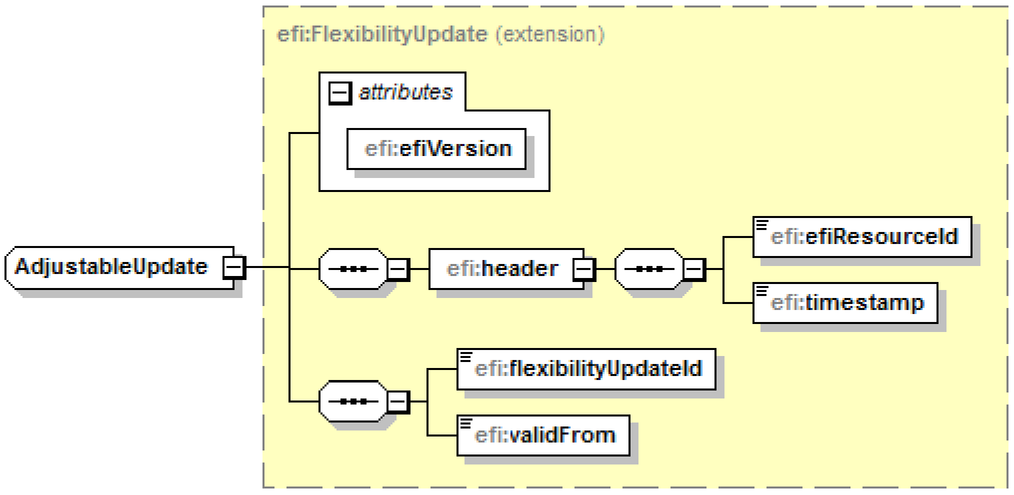

FIGURE 43: OVERVIEW OF THE ADJUSTABLEUPDATE CLASS

The AdjustableUpdate class is derived from the FlexibilityUpdate class (see 3.4) and does not feature additional attributes.

6.4 AdjustableSystemDescription

An adjustable device typically has a set of states it can assume (e.g. off, on, half-power, etc.) and has little restrictions for switching from one state to another. In S2 these states are called RunningModes. The AdjustableSystemDescription message describes the available RunningModes, possible transitions between them and the restrictions that are imposed on each transition.

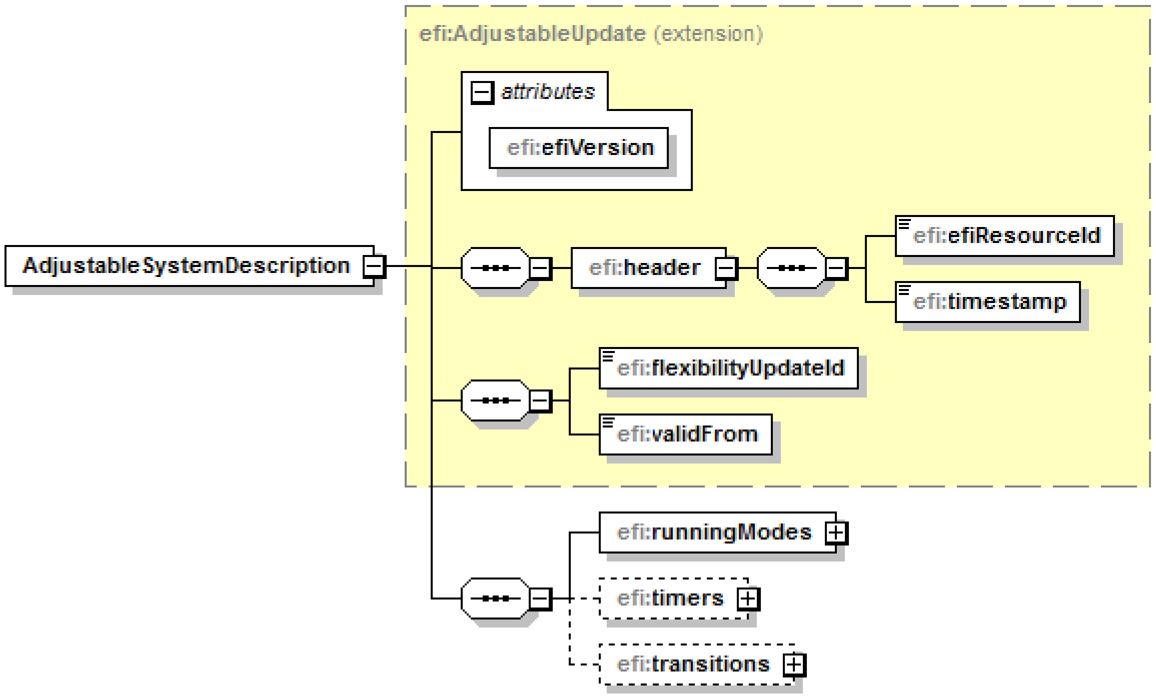

FIGURE 44: OVERVIEW OF THE ADJUSTABLESYSTEMDESCRIPTION CLASS

The AdjustableSystemDescription class is derived from the AdjustableUpdate class (see 6.3) and in addition features the following attributes:

Attribute	Description
runningModes (AdjustableRunningModes)	This attribute describes the runningModes for this AdjustableSystemDescription and is of the type AdjustableRunningModes. This type is explained in paragraph 6.4.2.
timers (Timers)	This optional attribute contains a list of timers. A timer can be used to restrict a transition. The type of this attribute is Timers and is explained in paragraph 6.4.6.
transitions (Transitions)	This attribute contains a list of transitions. A transition is the only way to go from one running mode to another. The type of this attribute is Transitions and is explained in paragraph 6.4.8.

TABLE 30: ATTRIBUTES OF THE ADJUSTABLESYSTEMDESCRIPTION CLASS

6.4.1 RunningMode

There are several RunningMode variations that are used within S2. This class is the parent class from which all variations are derived.

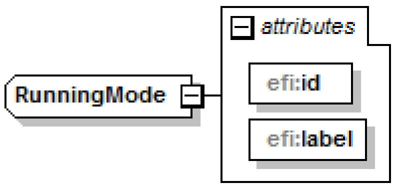

FIGURE 45: OVERVIEW OF THE RUNNINGMODE CLASS

The RunningMode class has the following attributes:

Attribute	Description
id (int)	This is a unique ID within the context of the AdjustableSystemDescription message to identify this RunningMode.
label (string)	This attribute is used to provide the RunningMode with a meaningful name such as “on” or “off”.

TABLE 31: ATTRIBUTES OF THE RUNNINGMODE CLASS

6.4.2 AdjustableRunningModes

This class contains a list of running modes, each entry of the list is either a discrete or a continuous running mode.

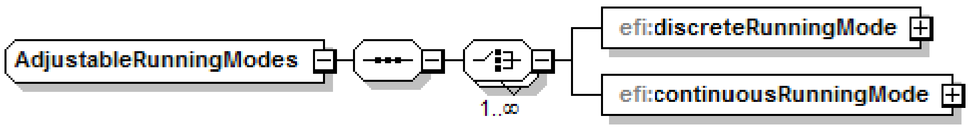

FIGURE 46: OVERVIEW OF THE ADJUSTABLERUNNINGMODES CLASS

The AdjustableRunningModes class has the following attributes:

Attribute	Description
discreteRunningMode (AdjustableDiscreteRunningMode)	This attribute is of the type AdjustableDiscreteRunningMode and is explained in paragraph 6.4.3.
continuousRunningMode (AdjustableContinuousRunningMode)	This attribute is of the type AdjustableContinuousRunningMode and is explained in paragraph 6.4.4.

TABLE 32: ATTRIBUTES OF THE ADJUSTABLERUNNINGMODES CLASS

6.4.3 AdjustableDiscreteRunningMode

Each running mode is associated with the consumption (negative consumption is production) of one or more commodities. This consumption can either be expressed as a range or a discrete level. This class expresses the discrete option.

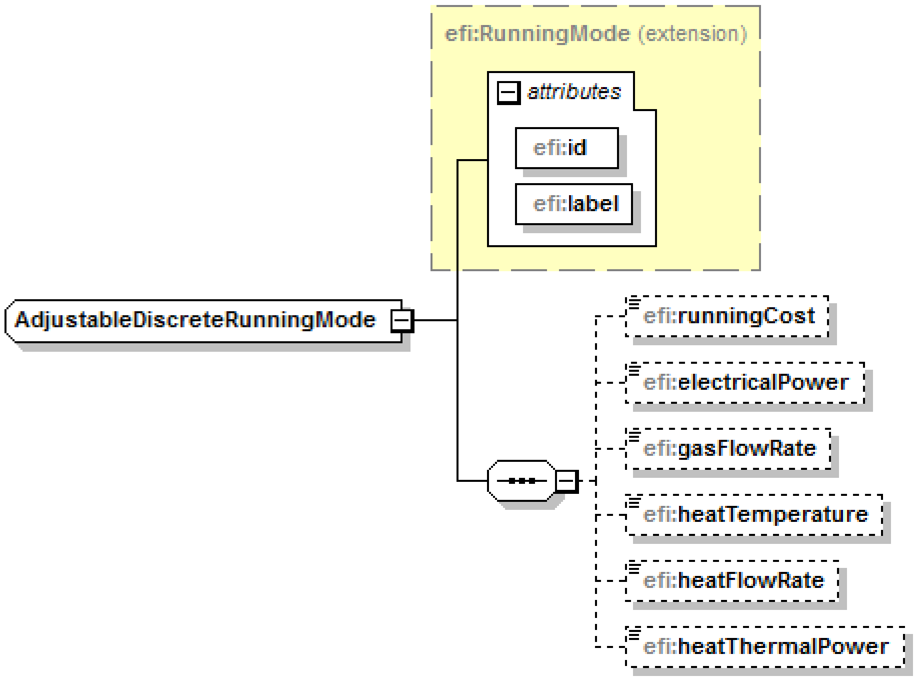

FIGURE 47: OVERVIEW OF THE ADJUSTABLEDISCRETERUNNINGMODE CLASS

The AdjustableDiscreteRunningMode is derived from the RunningMode class (see 6.4.1) and in addition features the following attributes:

Attribute	Description
runningCost (decimal)	This is optional attribute expresses the running costs (e.g. deprecation) that are associated with this RunningMode. The currency that is being used has been provided in the AdjustableRegistration message (see 6.2). These costs are expressed per second.
electricalPower (double)	This optional attribute is used to provide the electrical power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.
gasFlowRate (double)	This optional attribute is used to provide the gas consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatTemperature (double)	This optional attribute is used to provide the temperature of the heat commodity associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the degrees Celsius unit.

heatFlowRate (double)	This optional attribute is used to provide the flow rate of the heat commodity associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatThermalPower (double)	This optional attribute is used to provide the heat thermal power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.

TABLE 33: ATTRIBUTES OF THE ADJUSTABLEDISCRETERUNNINGMODE CLASS

6.4.4 AdjustableContinousRunningMode

Each running mode is associated with the consumption (negative consumption is production) of one or more commodities. This consumption can either be expressed as a range or a discrete level. This class expresses the range option.

A range has a lower and an upper bound, any value that lies between these bounds is valid. It is possible that multiple commodities are expressed in both bounds (e.g. electricityConsumption and heatConsumption). In that case there is a strict relation between the values that both commodities can assume within their respective ranges.

Consider an example where the electricity consumption has a lower bound of 1000 watt and an upper bound of 2000 watt. The heat consumption has a lower bound of -3000 thermal watt and an upper bound of -6000 watt. If the electricity consumption assumes a value of 1500 watt that is half way in its range, then the heat consumption must also be half way in its own range which corresponds to a value of -4500 thermal watt.

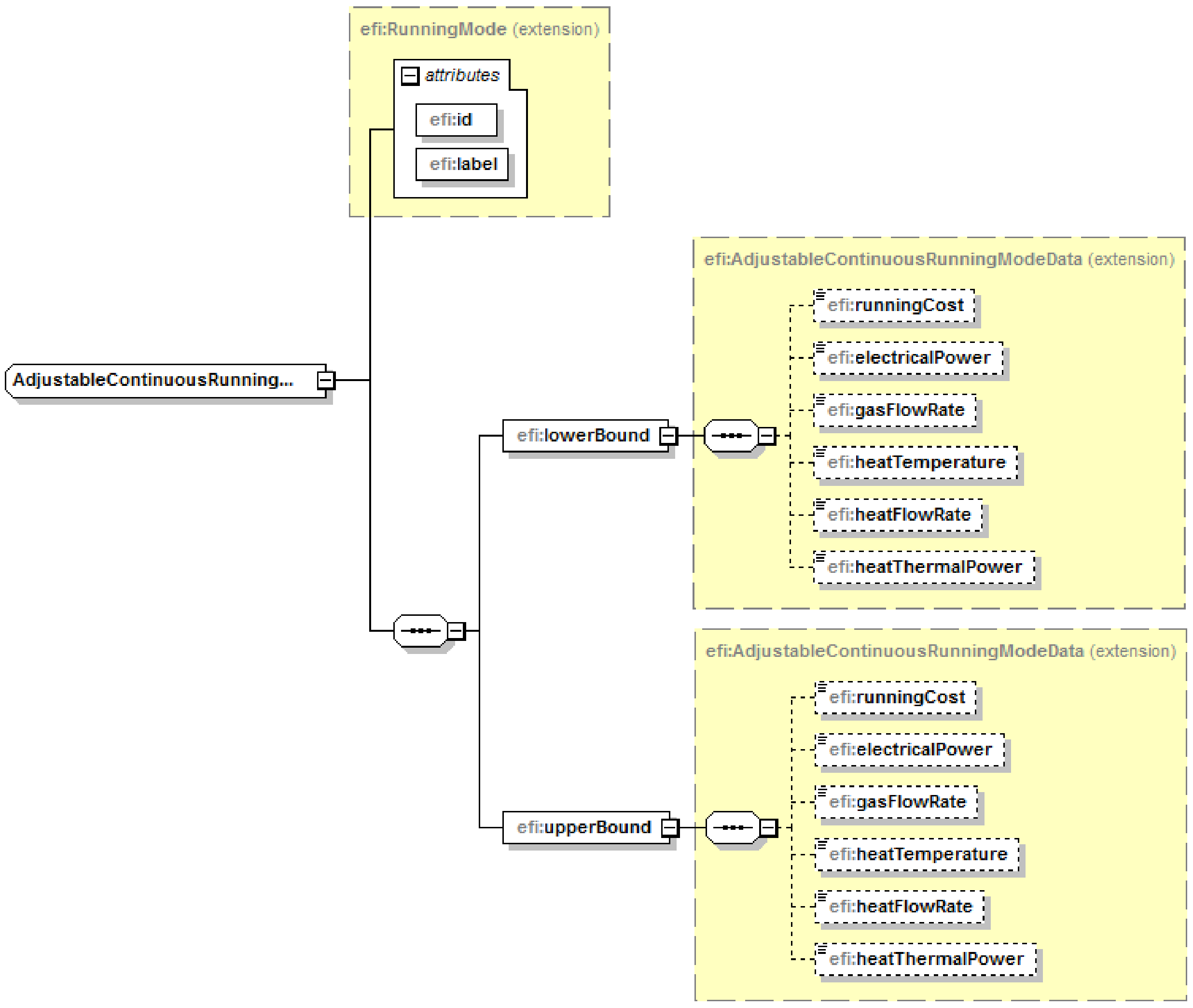

FIGURE 48: OVERVIEW OF THE ADJUSTABLECONTINUOUSRUNNINGMODE CLASS

The AdjustableContinuousRunningMode is derived from the RunningMode class (see 6.4.1) and in addition features the following attributes:

Attribute	Description
lowerBound (AdjustableContinuousRunningModeData)	This attribute provides all the information for the lower bound of this continuous running mode. The attribute type is derived from the AdjustableContinuousRunningModeData class which is explained in paragraph 6.4.5.
upperBound (AdjustableContinuousRunningModeData)	This attribute provides all the information for the upper bound of this continuous running mode. The attribute type is derived from the AdjustableContinuousRunningModeData class which is explained in paragraph 6.4.5.

TABLE 34: ATTRIBUTES OF THE ADJUSTABLECONTINUOUSRUNNINGMODE CLASS

6.4.5 AdjustableContinuousRunningModeData

This class contains all the information for the lower and upper bound of an AdjustableContinuousRunningMode class, it has the following attributes:

Attribute	Description
runningCost (decimal)	This is optional attribute expresses the running costs (e.g. deprecation) that are associated with this RunningMode. The currency that is being used has been provided in the AdjustableRegistration message (see 6.2). These costs are expressed per second.
electricalPower (double)	This optional attribute is used to provide the electrical power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.
gasFlowRate (double)	This optional attribute is used to provide the gas consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatTemperature (double)	This optional attribute is used to provide the temperature of the heat commodity associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the degrees Celsius unit.
heatFlowRate (double)	This optional attribute is used to provide the flow rate of the heat commodity associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatThermalPower (double)	This optional attribute is used to provide the heat thermal power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.

TABLE 35: ATTRIBUTES OF THE ADJUSTABLECONTINUOUSRUNNINGMODEDATA CLASS

6.4.6 Timers

Timers can be used to place restrictions on the transition from one running mode to another. This class contains a list of all available timers.

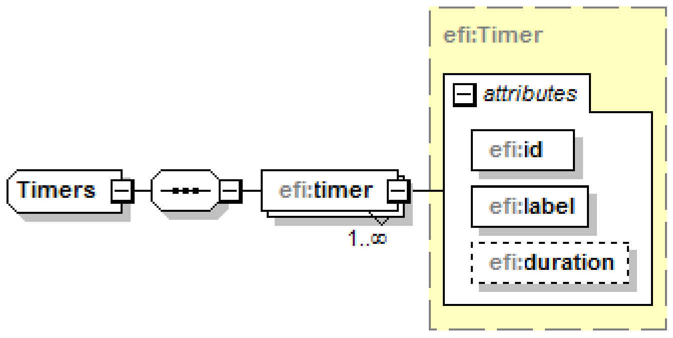

FIGURE 49: OVERVIEW OF THE TIMERS CLASS

The Timers class has the following attributes:

Attribute	Description
timer (Timer)	This attribute describes a single timer. There can be multiple timer attributes within the Timers class. The Timer type is explained in paragraph 6.4.7.

TABLE 36: ATTRIBUTES OF THE TIMERS CLASS

6.4.7 Timer

This class represents the concept of a countdown timer. When the timer is being “set” it assumes the duration value (e.g. 5 minutes) and starts to count down until it reaches zero. The Timer class has the following attributes:

Attribute	Description
id (int)	This is a unique ID within the context of the AdjustableSystemDescription message to identify this Timer.
label (string)	This attribute is used to provide the Timer with a meaningful name such as “minimum on time” or “minimum off time”.
duration (duration)	This attribute represents the duration of this countdown timer. When the timer is “set” it will assume this value and start to count down to zero.

TABLE 37: ATTRIBUTES OF THE TIMER CLASS

6.4.8 Transitions

A transition is the only way to go from one running mode to another.

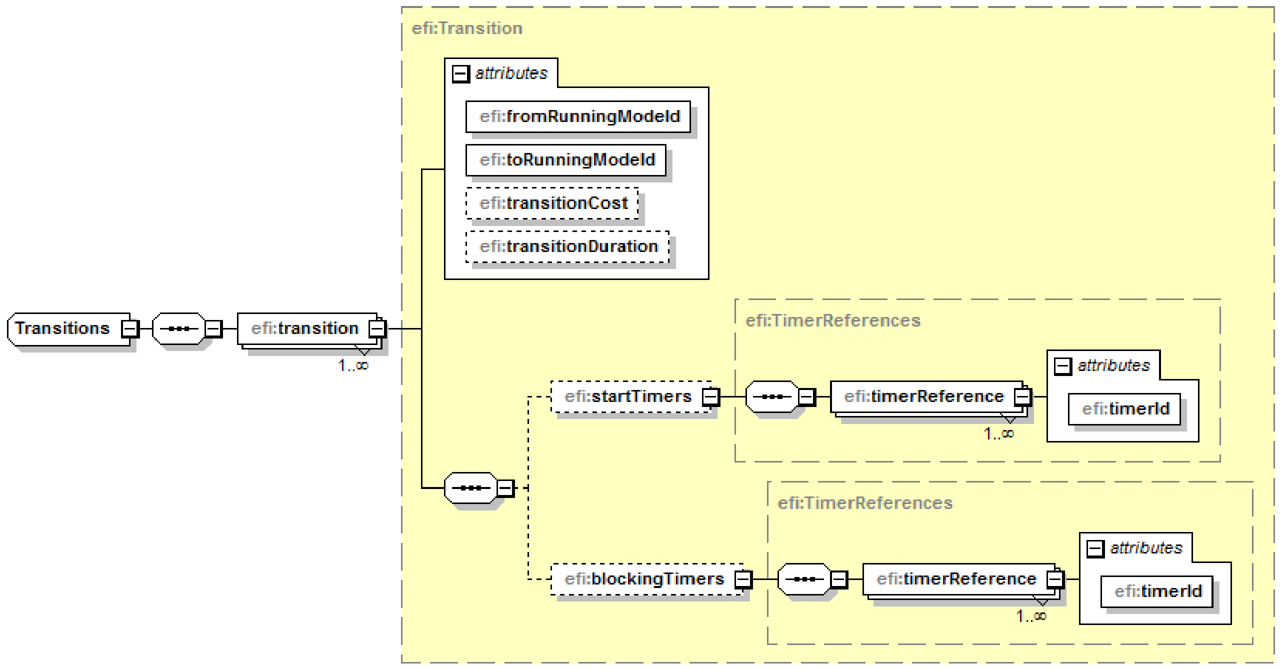

FIGURE 50: OVERVIEW OF THE TRANSITIONS CLASS

The Transitions class has the following attributes:

Attribute	Description
transition (Transition)	This attribute describes a single transition. There can be multiple transition attributes within the Transitions class. The Transition type is explained in paragraph 6.4.9.

TABLE 38: ATTRIBUTES OF THE TRANSITIONS CLASS

6.4.9 Transition

The Transition class has the following attributes:

Attribute	Description
fromRunningModeId (int)	This attributes refers to the ID of the running mode that this Transition starts from.
toRunningModeId (int)	This attributes refers to the ID of the running mode that this Transition is moving towards.
transitionCost (decimal)	This is optional attribute expresses the costs (e.g. deprecation) that are associated with this transition. The currency that is being used has been provided in the AdjustableRegistration message (see 6.2).
transitionDuration (duration)	This optional attribute indicates how much time it will

	take to fully complete the transition between the involved running modes. The transition duration can be used to approximate ramp up/down characterics.
startTimers (TimerReferences)	This attribute contains references to timers that have to be started (or set) when this transition is initiated. A typical example would be to start a “minimum on” timer when switching to the “on” running mode to prevent the device from being switched off prematurely. This attribute is of the type TimerReferences and is explained in paragraph 6.4.10.
blockingTimers (TimerReferences)	This attribute contains references to timers that have to be finished (completed a countdown to zero) before this transition can be initiated. These timers block the transition as long as at least one of them has a non-zero value. A typical example would be to check whether the “minimum on” timer has finished before switching to the “off” running mode to prevent the device from being switched off prematurely. This attribute is of the type TimerReferences and is explained in paragraph 6.4.10.

TABLE 39: ATTRIBUTES OF THE TRANSITION CLASS

6.4.10 TimerReferences

The TimerReferences class has the following attributes:

Attribute	Description
timerReference (TimerReference)	This attribute describes a single timerReference. There can be multiple timerReference attributes within the TimerReferences class. The TimerReference type is explained in paragraph 6.4.11.

TABLE 40: ATTRIBUTES OF THE TIMERREFERENCES CLASS

6.4.11 TimerReference

The TimerReference class has the following attributes:

Attribute	Description
timerId (int)	This attribute refers to the ID of a Timer object.

TABLE 41: ATTRIBUTES OF THE TIMERREFERENCE CLASS

6.5 AdjustableStatus

The system description contains a “static” model of an adjustable device. The CEM needs to combine this knowledge with the current status of the device in order to come up with valid instructions. The RM will periodically send an AdjustableStatus message to provide the CEM with this necessary piece of information.

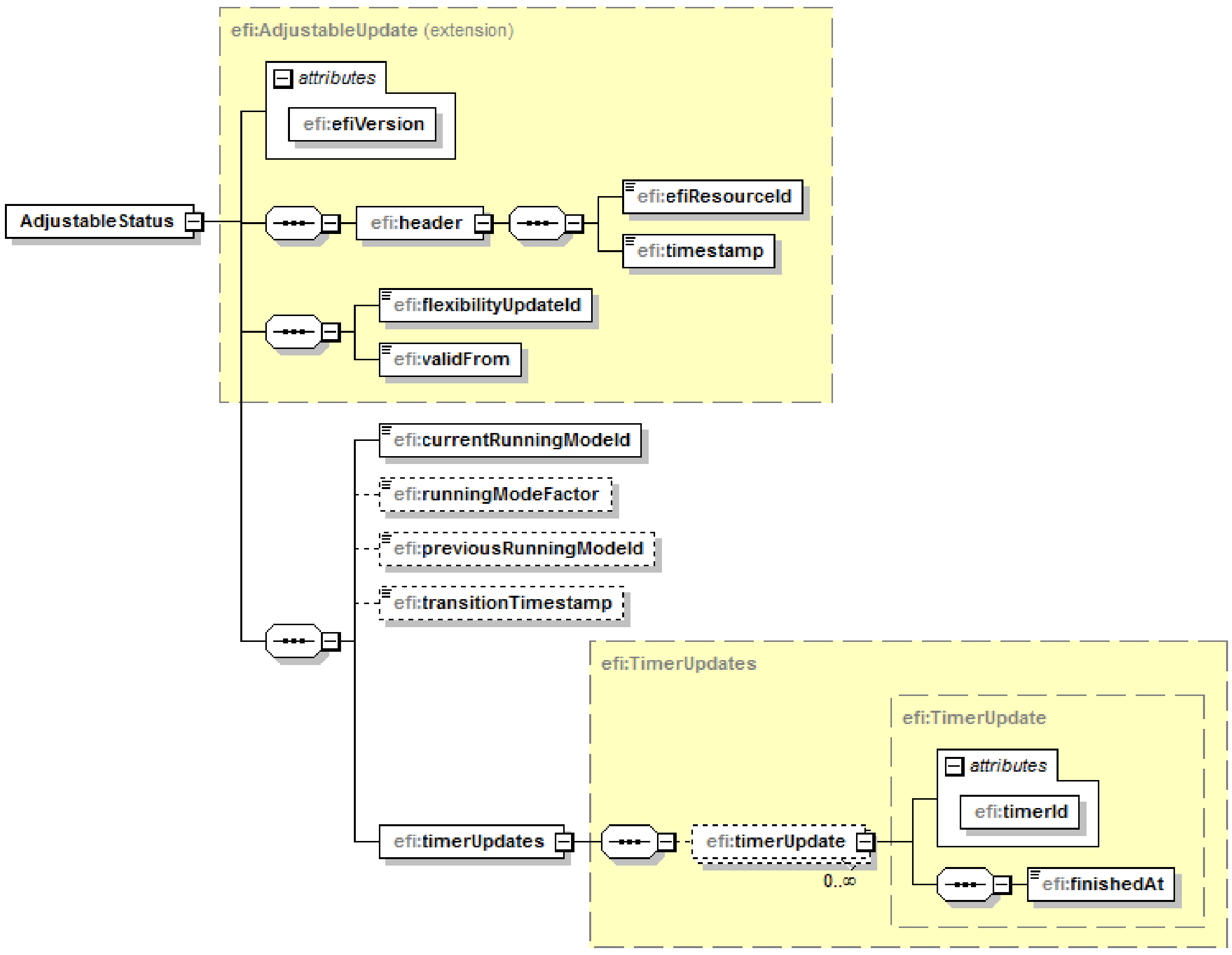

FIGURE 51: OVERVIEW OF THE ADJUSTABLESTATUS CLASS

The AdjustableStatus class is derived from the AdjustableUpdate class (see 6.3) and in addition features the following attributes:

Attribute	Description
currentRunningModeId (int)	This attribute refers to the ID of the running mode that is currently selected.
runningModeFactor (double)	This attribute is optional, but MUST be used when the currentRunningModeId refers to a continuous running mode. The factor must be a number between 0 and 1. When it is 0 the lower bound value must be used for all involved commodities, the upper bound value must be used when this factor is 1. If the factor is between 0 and 1, then that portion of the range should be added to the

	lower bound values.
previousRunningModeId (int)	This attribute is optional, but MUST be provided by the Resource Manager when this information is available. It refers to the ID of the running mode that was selected before the current running mode.
transitionTimestamp (dateTime)	This attribute is optional, but MUST be provided by the Resource Manager when this information is available. It refers to the point in time when the switch to the current running mode was made. If applicable the CEM can combine this with the transitionDuration information provided in the AdjustableSystemDescription message to derive whether the device might still be ramping up or down.
timerUpdates (TimerUpdates)	This attribute contains a list of timer updates. The type of the attribute is TimerUpdates which is explained in paragraph 6.5.1.

TABLE 42: ATTRIBUTES OF THE ADJUSTABLESTATUS CLASS

6.5.1 TimerUpdates

The TimerUpdates class has the following attributes:

Attribute	Description
timerUpdate (TimerUpdate)	This attribute describes a single timerUpdate. There can be multiple timerUpdate attributes within the TimerUpdates class. The TimerUpdate type is explained in paragraph 6.5.2.

TABLE 43: ATTRIBUTES OF THE TIMERUPDATES CLASS

6.5.2 TimerUpdate

The TimerUpdate class has the following attributes:

Attribute	Description
timerId (int)	This attribute refers to the ID of the timer that this timer update has information about.
finishedAt (dateTime)	This attributed indicates the time at which point the referenced timer will have completed its countdown to zero.

TABLE 44: ATTRIBUTES OF THE TIMERUPDATE CLASS

6.5.3 AdjustableInstruction

This instruction message is being sent from the CEM to the RM. It contains the selection of a particular running mode as well as the desired switch time.

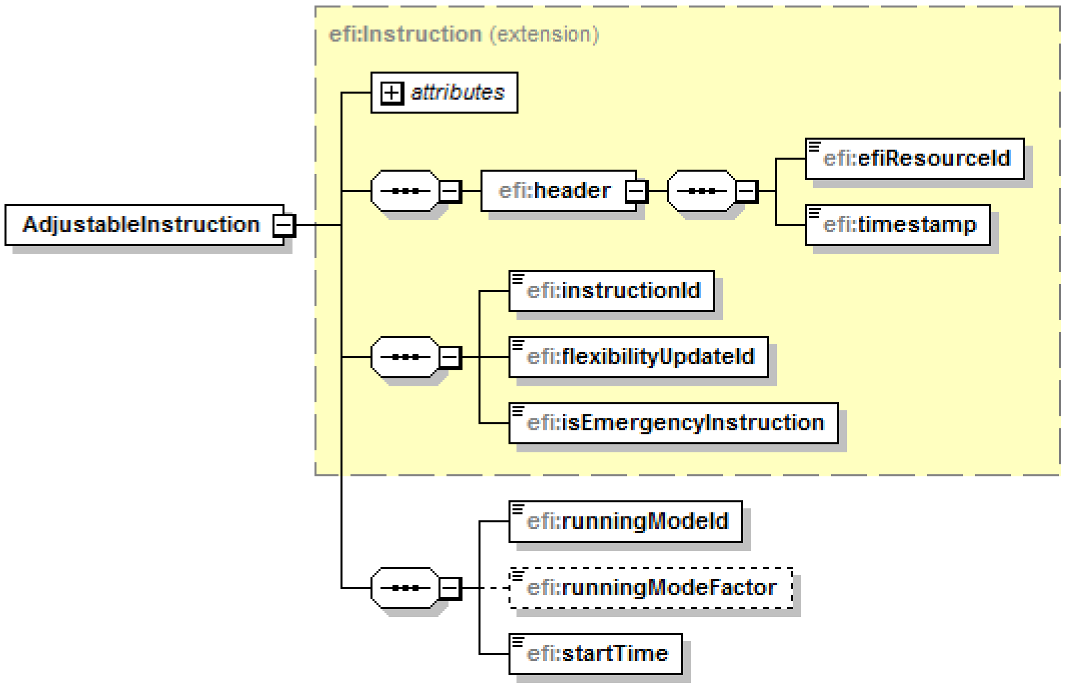

FIGURE 52: OVERVIEW OF THE ADJUSTABLEINSTRUCTION CLASS

The AdjustableInstruction class is derived from Instruction (see section 3.5) and in addition features the following attributes:

Attribute	Description
runningModeId (int)	This attribute refers to the ID of the running mode that the CEM would like the RM to switch to.
runningModeFactor (double)	This attribute is optional, but MUST be used when the runningModeId refers to a continuous running mode. The factor must be a number between 0 and 1. When it is 0 the lower bound value must be used for all involved commodities, the upper bound value must be used when this factor is 1. If the factor is between 0 and 1, then that portion of the range should be added to the lower bound value.
startTime (dateTime)	The point in time at which this instruction should be executed by the RM

TABLE 45: ATTRIBUTES OF THE ADJUSTABLEINSTRUCTION CLASS

7 Storage category

Storage devices are devices that can be flexible in production/consumption but are restricted by a buffer

7.1 Typical Storage Message Flow

shows the message flow of the Storage category. Please note that all messages are derived from the generic messages shown in Figure 4.

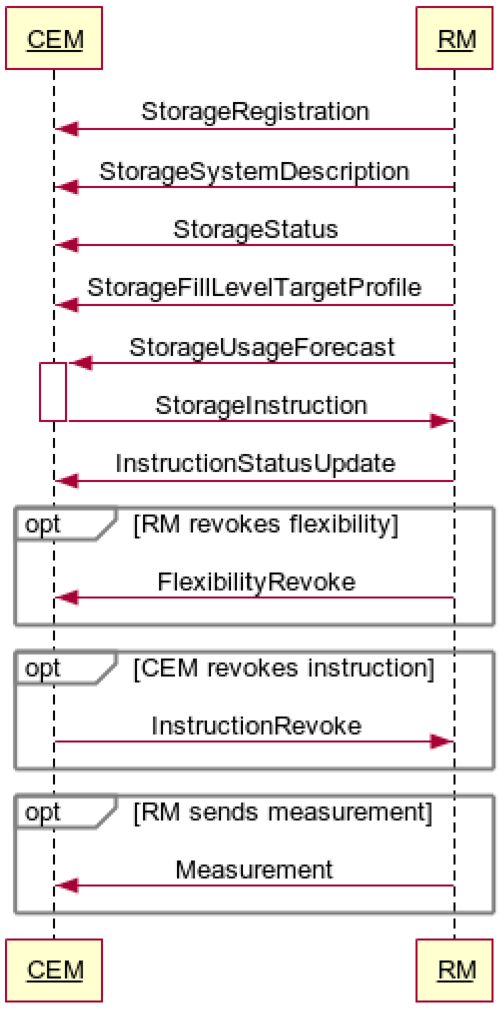

FIGURE 53: A TYPICAL STORAGE MESSAGE FLOW

In the following paragraphs each message is described in detail, with the exception of the messages that have already been described in previous chapters: InstructionStatusUpdate (see 3.6), FlexibilityRevoke (see 3.7), InstructionRevoke (see 3.8) and Measurement (see 3.9).

7.2 StorageRegistration

This message is used by Resource Managers that represent a storage device to inform the CEM of the capabilities of that device.

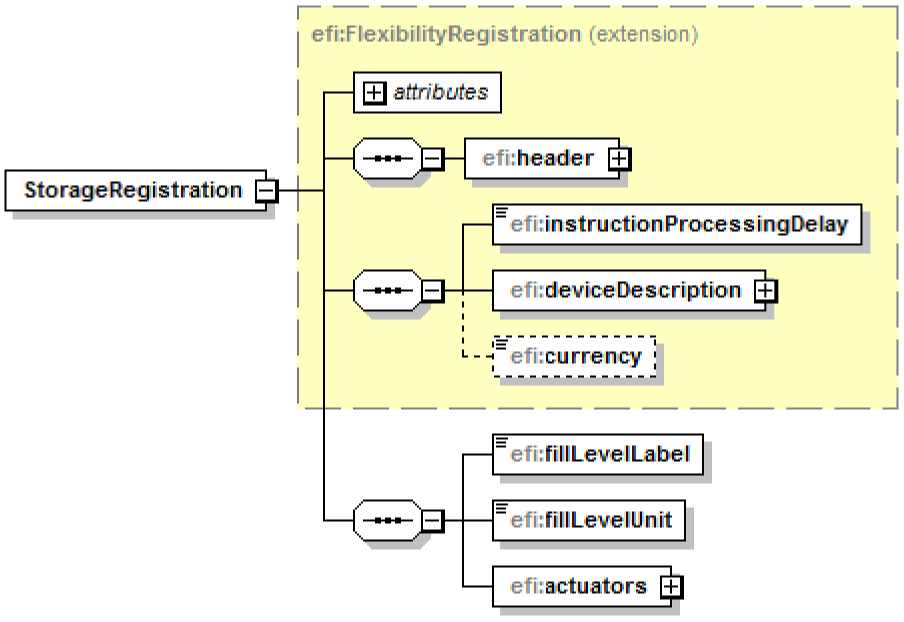

FIGURE 54: OVERVIEW OF THE STORAGEREGISTRATION CLASS

The StorageRegistration class is derived from the FlexibilityRegistration (see section 3.3) and in addition features the following attributes:

Attribute	Description
fillLevelLabel (string)	This attribute of the type string is purely informative. It gives the end-user an impression of what is being stored in this storage device. Examples of such a label are: “boiler temperature”, “battery charge level”, etc.
fillLevelUnit (string)	This attribute of the type string is purely informative. It gives the end-user an impression of the unit in which the fill level is expressed, scuh as “temperature”, “kWh”, “percentage”. To the CEM the fillLevelUnit and fillLevelLabel are not relevant.
actuators (Actuators)	This attribute contains a list of actuators that this storage device has. It is of the type Actuators and is further explained in paragraph 7.2.1.

TABLE 46: ATTRIBUTES OF THE STORAGEREGISTRATION CLASS

7.2.1 Actuators

Actuators can convert one or more commodities (electricity, gas or heat) into the quantity that is being stored in this storage device or vice versa. A simple example of such an actuator would be a heating coil converting electricity into hot water. Figure 55 provides an overview of the Actuators class.

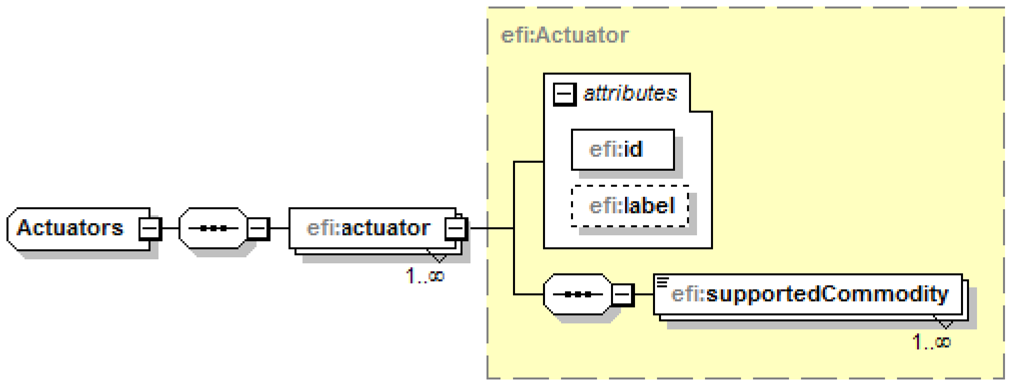

FIGURE 55: OVERVIEW OF THE ACTUATORS CLASS

The Actuators class has the following attributes:

Attribute	Description
actuator (Actuator)	This attribute describes a single actuator. There can be multiple actuator attributes within the TimerUpdates class. The Actuator type is explained in paragraph .

FIGURE 56: ATTRIBUTES OF THE ACTUATORS CLASS

7.2.2 Actuator

The Actuator class only contains static information about the actuator. Its behaviour which might change over time is described in subsequent messages. This class has the following attributes:

Attribute	Description
id (int)	This is a unique ID within the context of this storage device to identify this actuator.
label (string)	This optional attribute may be used to provide a meaningful label for this actuator, e.g. “Heating coil”.
supportedCommodity (CommodityEnum)	The attribute commodityType indicates which commodities are supported by this actuator. It is possible to specify multiple commodityTypes. This attribute is of the type CommodityEnum which is further explained in paragraph 5.2.1.

TABLE 47: ATTRIBUTES OF THE ACTUATOR CLASS

7.3 StorageUpdate

There are multiple update messages within the Storage class, they are all derived from the StorageUpdate class which is shown in Figure 57.

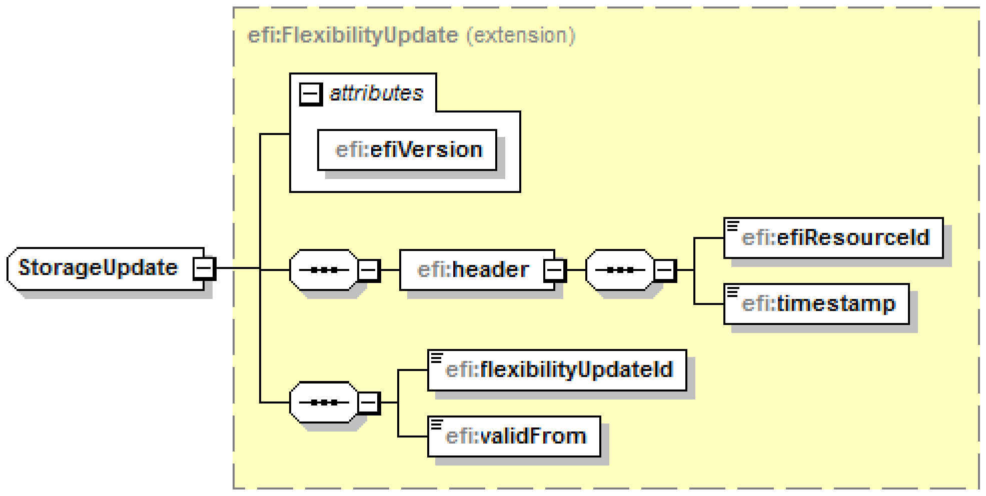

FIGURE 57: OVERVIEW OF THE STORAGEUPDATE CLASS

The StorageUpdate class is derived from the FlexibilityUpdate class (see 3.4) and does not feature additional attributes.

7.4 StorageSystemDescription

An storage device typically has a set of states it can assume (e.g. off, on, half-power, etc.. The possible states are limited by the fill level of the storage of the device. If the storage is (almost) full for instance, it will not be possible to switch the device to full power. In S2 these states are called RunningModes. The StorageSystemDescription message describes the available RunningModes, possible transitions between them and the restrictions that are imposed on each transition.

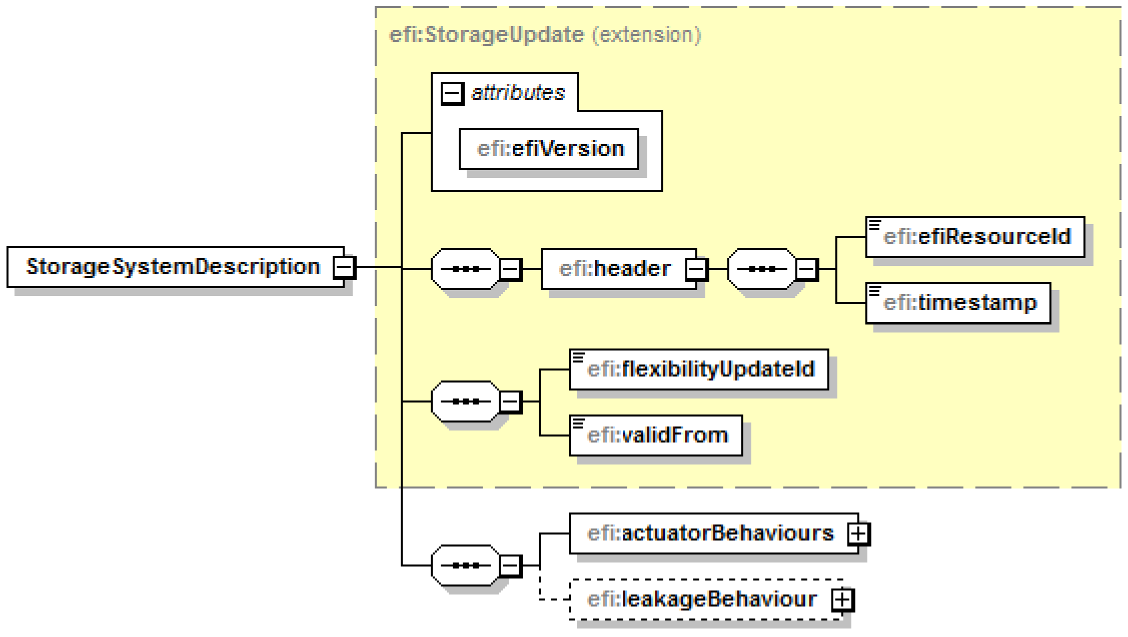

FIGURE 58: OVERVIEW OF THE STORAGESYSTEMDESCRIPTION CLASS

The StorageSystemDescription class is derived from the StorageUpdate class (see 7.3) and in addition features the following attributes:

Attribute	Description
actuatorBehaviours (ActuatorBehaviours)	This attribute contains a list of actuator behaviours. The actuator behaviour describes in detail how the actuator interacts with the storage. The type of the attribute is ActuatorBehaviours and is further explained in paragraph 7.4.3.
leakageBehaviour (LeakageFunction)	Almost all storage devices will experience leakage to some extent. This attribute describes the leakage behaviour of this storage device. The type of this attribute is LeakageFunction and is further explained in paragraph 7.4.1.

TABLE 48: ATTRIBUTES OF THE STORAGESYSTEMDESCRIPTION CLASS

7.4.1 LeakageFunction

Almost all storage devices will experience leakage to some extent. In order to understand the constraints of a storage device it is important to know how the leakage behaves. This class describes the leakage function; it describes the relation between fill level and leakage rate.

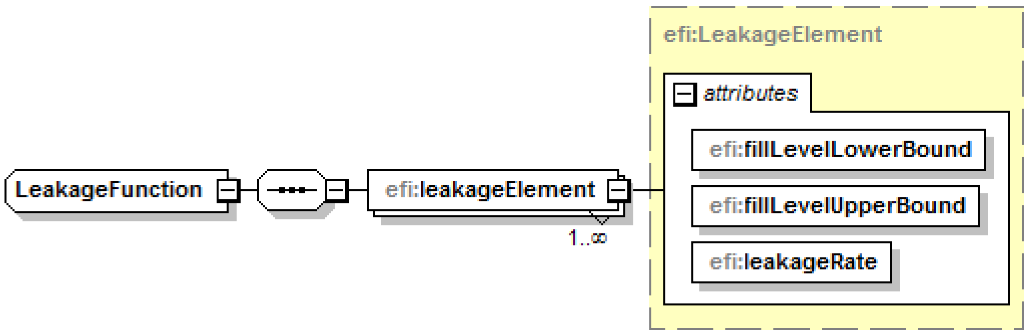

FIGURE 59: OVERVIEW OF THE LEAKAGEFUNCTION CLASS

The LeakageFunction has the following attributes:

Attribute	Description
leakageElement (LeakageElement)	This attribute describes a single leakage element. There can be multiple leakage element attributes within the LeakageFunction class. The LeakageElement type is explained in paragraph 7.4.2.

TABLE 49: ATTRIBUTES OF THE LEAKAGEFUNCTION CLASS

7.4.2 LeakageElement

This class contains a fill level range and a leakage rate. An example would be a hot water storage with a leakage rate of 0.00027 degrees Celsius per second (around 1 degree Celsius per hour) when the fill level is between 70 and 80 degrees Celsius. Although this example shows the unit (degrees Celsius) of what is being stored, it is not necessary to know about the unit, only the numbers themselves are relevant. The LeakageElement class has the following attributes:

Attribute	Description
fillLevelLowerBound (double)	This attribute is the lower bound of the fill level range in which the leakage rate is valid. It is of the type double.
fillLevelUpperBound (double)	This attribute is the upper bound of the fill level range in which the leakage rate is valid. It is of the type double.
leakageRate (double)	This attribute describes the leakage rate. It must always be interpreted to be a rate that is expressed per second. If the leakage rate is positive (e.g. in the case of hot water storage) and the device is idle, the fill level will come down over time. If the leakage rate is negative (e.g. in the case of a freezer) and the device is idle the fill level will go up over time.

TABLE 50: ATTRIBUTES OF THE LEAKAGEELEMENT CLASS

7.4.3 ActuatorBehaviours

This class describes how each of the actuators (e.g. heat coil, gas burner, etc.) in this StorageSystemDescription message interact with the storage.

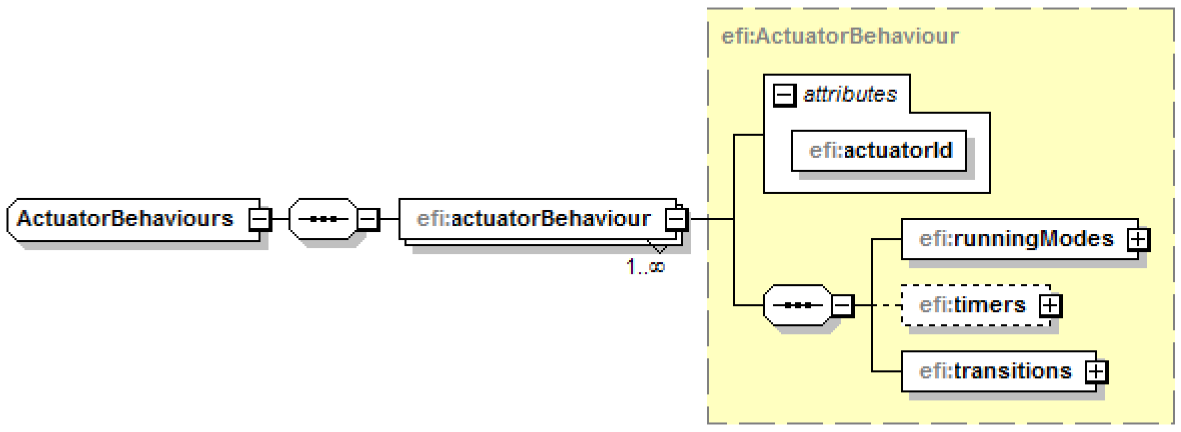

FIGURE 60: OVERVIEW OF THE ACTUATORBEHAVIOURS CLASS

The ActuatorBehaviours class has the following attributes:

Attribute	Description

actuatorBehaviour (ActuatorBehaviour)	This attribute describes a single actuator behaviour. There can be actuator behaviour attributes within the ActuatorBehaviours class. The ActuatorBehaviour type is explained in paragraph 7.4.4.

TABLE 51: ATTRIBUTES OF THE ACTUATORBEHAVIOURS CLASS

7.4.4 ActuatorBehaviour

The ActuatorBehaviour class has the following attributes:

Attribute	Description
actuatorId (int)	This is a unique ID within the context of this storage device to identify this actuator. It references the actuator that was declared in the StorageRegistration message.
runningModes (StorageRunningModes)	This attribute describes the runningModes for the actuator identified by the above actuatorId. It of the type StorageRunningModes and is explained in paragraph 6.4.2.
timers (Timers)	The optional timers attribute is of the type Timers and is explained in 6.4.6.
transitions (Transitions)	The transitions attribute is of the type Transitions and is further explained in 6.4.8.

TABLE 52: ATTRIBUTES OF THE ACTUATORBEHAVIOUR CLASS

7.4.5 StorageRunningModes

This class contains a list of running modes, each entry of the list is either a discrete or a continuous running mode.

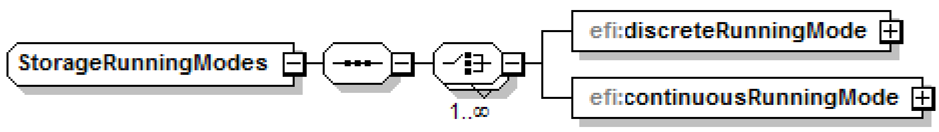

FIGURE 61: OVERVIEW OF THE STORAGERUNNINGMODES CLASS

The StorageRunningModes class has the following attributes:

Attribute	Description
discreteRunningMode (StorageDiscreteRunningMode)	This attribute is of the type StorageDiscreteRunningMode and is explained in paragraph 7.4.6.
continuousRunningMode	This attribute is of the type

(StorageContinuousRunningMode)	StorageContinuousRunningMode and is explained in paragraph 7.4.9.

TABLE 53: ATTRIBUTES OF THE STORAGERUNNINGMODES CLASS

7.4.6 StorageDiscreteRunningMode

There are two types of running modes for storage devices; the discrete and the continuous variant. This class expresses the discrete option.

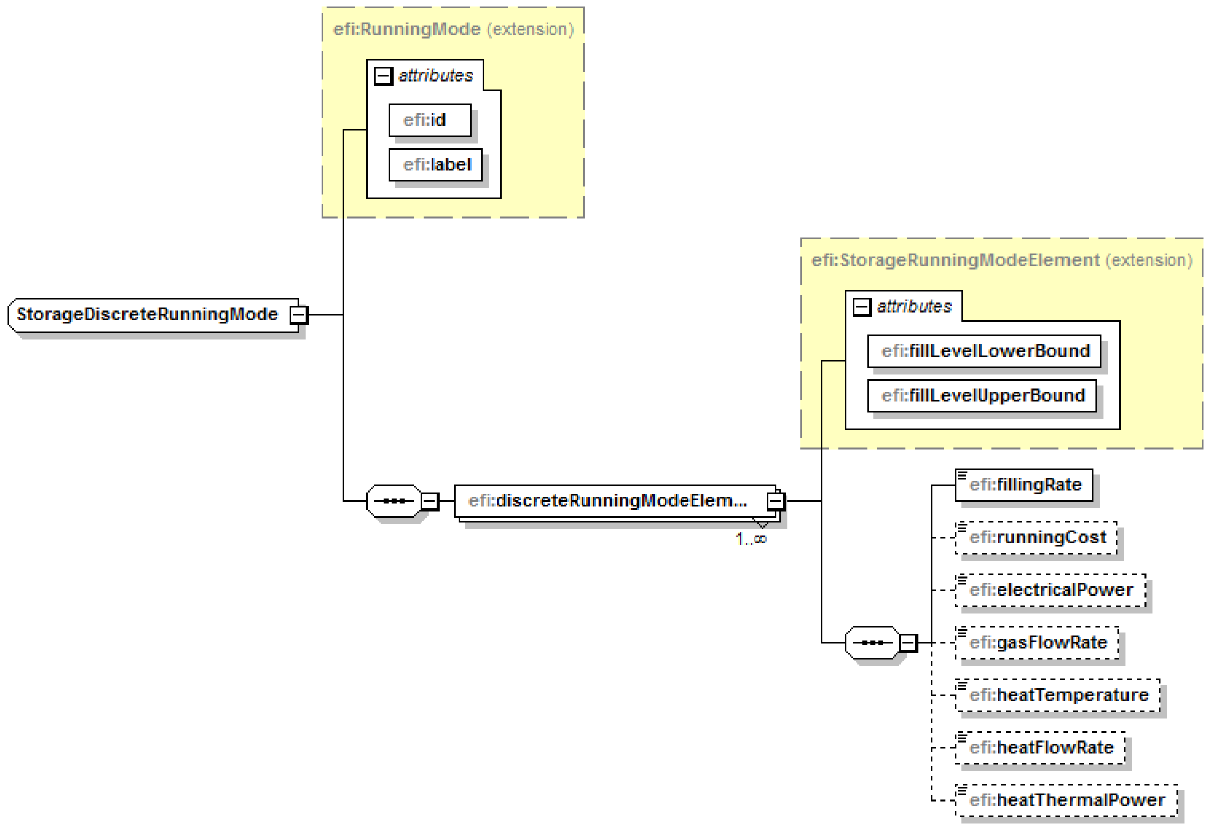

FIGURE 62: OVERVIEW OF THE STORAGEDISCRETERUNNINGMODE CLASS

The StorageDiscreteRunningMode class is derived from the RunningMode class (see 6.4.1) and in addition features the following attributes:

Attribute	Description
discreteRunningModeElement (DiscreteRunningModeElement)	This attribute is of the type DiscreteRunningModeElement and is explained in paragraph 7.4.8.

TABLE 54: ATTRIBUTES OF THE STORAGEDISCRETERUNNINGMODE CLASS

7.4.7 StorageRunningModeElement

A running mode describes the relation between the filling rate of the storage and the consumption (negative consumption is production) of one or more commodities. In most cases the filling rate and/or the commodity consumption will not be the consistent across the entire fill level range. When filling a hot water buffer with a heat coil on full power for instance, the filling rate will be higher when the fill level (in this case temperature of the hot water storage) is low. The higher the fill level, the harder it becomes to raise the temperature of the hot water storage.

In order to approximate the nonlinear nature of the fill level function, the running mode contains running mode elements that each describe a section of the entire fill level range. This parent class is used to describe a single section, it has the following attributes:

Attribute	Description
fillLevelLowerBound (double)	This attribute is the lower bound of the fill level range of this StorageRunningModeElement and is of the type double.
fillLevelUpperBound (double)	This attribute is the upper bound of the fill level range of this StorageRunningModeElement and is of the type double.

TABLE 55: ATTRIBUTES OF THE STORAGERUNNINGMODEELEMENT CLASS

7.4.8 DiscreteRunningModeElement

The DiscreteRunningModeElement is derived from the StorageRunningModeElement class (see 7.4.7) and in addition features the following attributes:

Attribute	Description
fillingRate (double)	This attribute expresses the filling rate for this DiscreteRunningModeElement. The value that is provided here must be interpreted as value/second.
runningCost (double)	This is optional attribute expresses the running costs (e.g. deprecation) that are associated with this DiscreteRunningModeElement. The currency that is being used has been provided in the AdjustableRegistration message (see 7.2). These costs are expressed per second.
electricalPower (double)	This optional attribute is used to provide the electrical power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.
gasFlowRate (double)	This optional attribute is used to provide the gas consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatTemperature (double)	This optional attribute is used to provide the temperature of the heat commodity associated with this

	DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the degrees Celsius unit.
heatFlowRate (double)	This optional attribute is used to provide the flow rate of the heat commodity associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatThermalPower (double)	This optional attribute is used to provide the heat thermal power consumption associated with this DiscreteRunningModeElement. The type of this attribute is double. This double must be interpreted as having the watt unit.

TABLE 56: ATTRIBUTES OF THE DISCRETERUNNINGMODEELEMENT CLASS

7.4.9 StorageContinousRunningMode

The discrete running mode element only allows single commodity consumption and filling rate values for each fill level range. This is not always practical. Consider a battery that is capable of charging and discharging between 3kW and -3kW in steps of 1 watt. When this would have to be modelled with the use of the StorageDiscreteRunningMode class, one would end up with 6000 running modes.

The StorageContinousRunningMode allows for ranges to be specified for commodity consumption and filling rate values. This way the number of running modes needed to properly model the example above can remain limited.

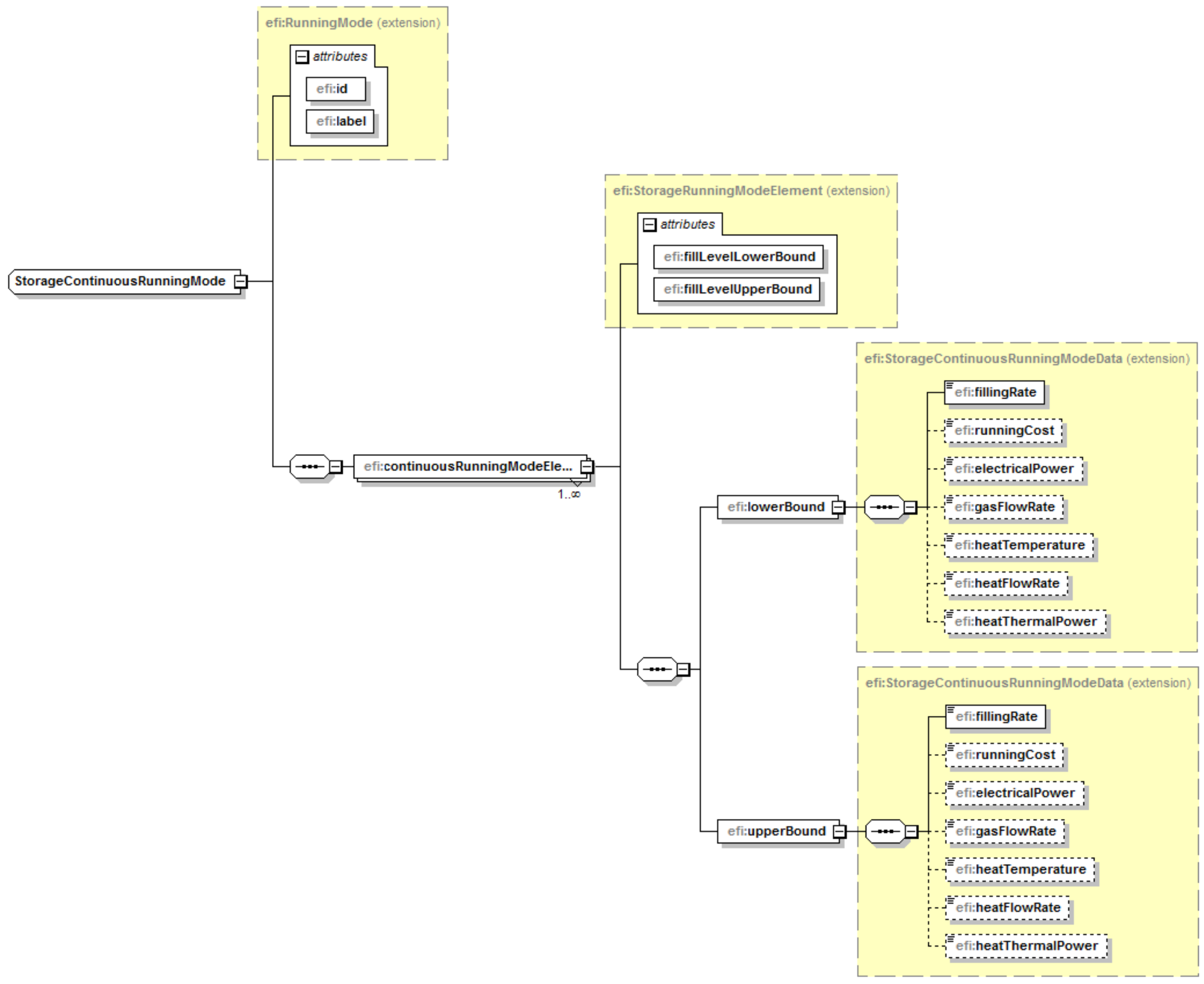

FIGURE 63: OVERVIEW OF THE STORAGECONTINUOUSRUNNINGMODE CLASS

The StorageContinuousRunningMode class is derived from the RunningMode class (see 6.4.1) and in addition features the following attributes:

Attribute	Description
continuousRunningModeElement (ContinuousRunningModeElement)	This attribute is of the type ContinuousRunningModeElement and is explained in paragraph 7.4.10.

TABLE 57: ATTRIBUTES OF THE STORAGECONTINUOUSRUNNINGMODE CLASS

7.4.10 ContinuousRunningModeElement

The ContinuousRunningModeElement is derived from the StorageRunningModeElement class (see 7.4.7). Where the DiscreteRunningModeElement only provides single values for the filling rate and each commodity consumption, this class allows one to define a lower and upper bound for each value.

An example would be a running mode element that has lower bound values of 0.01 for the filling rate and 500 watt of electricity power. The respective upper bound values are 0.02 and 1100 watt. If the device is being operated half way between the lower and upper bounds in this particular running mode element then the filling rate will be 0.015 whereas the electricity power consumed will amount to 800 watt.

The ContinuousRunningModeElement is derived from the StorageRunningModeElement class (see 7.4.7). and in addition features the following attributes:

Attribute	Description
lowerBound (StorageContinuousRunningModeData)	This attribute expresses all lower bound values that are associated with this ContinousRunningModeElement. It is of the type StorageContinousRunningModeData and is explained in paragraph .
upperBound (StorageContinuousRunningModeData)	This attribute expresses all upper bound values that are associated with this ContinousRunningModeElement. It is of the type StorageContinousRunningModeData and is explained in paragraph .

TABLE 58: ATTRIBUTES OF THE CONTINUOUSRUNNINGMODEELEMENT CLASS

7.4.11 StorageContinuousRunningModeData

This class contains all values that can be part of a lower and upper bound of a ContinuousRunningModeElement. It has the following attributes:

Attribute	Description
fillingRate (double)	This attribute expresses the filling rate for this lower/upper bound. The value that is provided here must be interpreted as value/second.
runningCost (double)	This is optional attribute expresses the running costs (e.g. deprecation) that are associated with this lower/upper bound. The currency that is being used has been provided in the AdjustableRegistration message (see 7.2). These costs are expressed per second.
electricalPower (double)	This optional attribute is used to provide the electrical power consumption associated with this lower/upper bound. The type of this attribute is double. This double must be interpreted as having the watt unit.
gasFlowRate (double)	This optional attribute is used to provide the gas consumption associated with this lower/upper bound. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatTemperature (double)	This optional attribute is used to provide the temperature of the heat commodity associated with this lower/upper bound. The type of this attribute is double. This double

	must be interpreted as having the degrees Celsius unit.
heatFlowRate (double)	This optional attribute is used to provide the flow rate of the heat commodity associated with this lower/upper bound. The type of this attribute is double. This double must be interpreted as having the m^3/s unit.
heatThermalPower (double)	This optional attribute is used to provide the heat thermal power consumption associated with this lower/upper bound. The type of this attribute is double. This double must be interpreted as having the watt unit.

TABLE 59: ATTRIBUTES OF THE STORAGECONTINUOUSRUNNINGMODEDATA CLASS

7.5 StorageStatus

The system description contains a “static” model of a storage device. The CEM needs to combine this knowledge with the current status of the device in order to come up with valid instructions. The RM will periodically send an StorageStatus message to provide the CEM with this necessary piece of information.

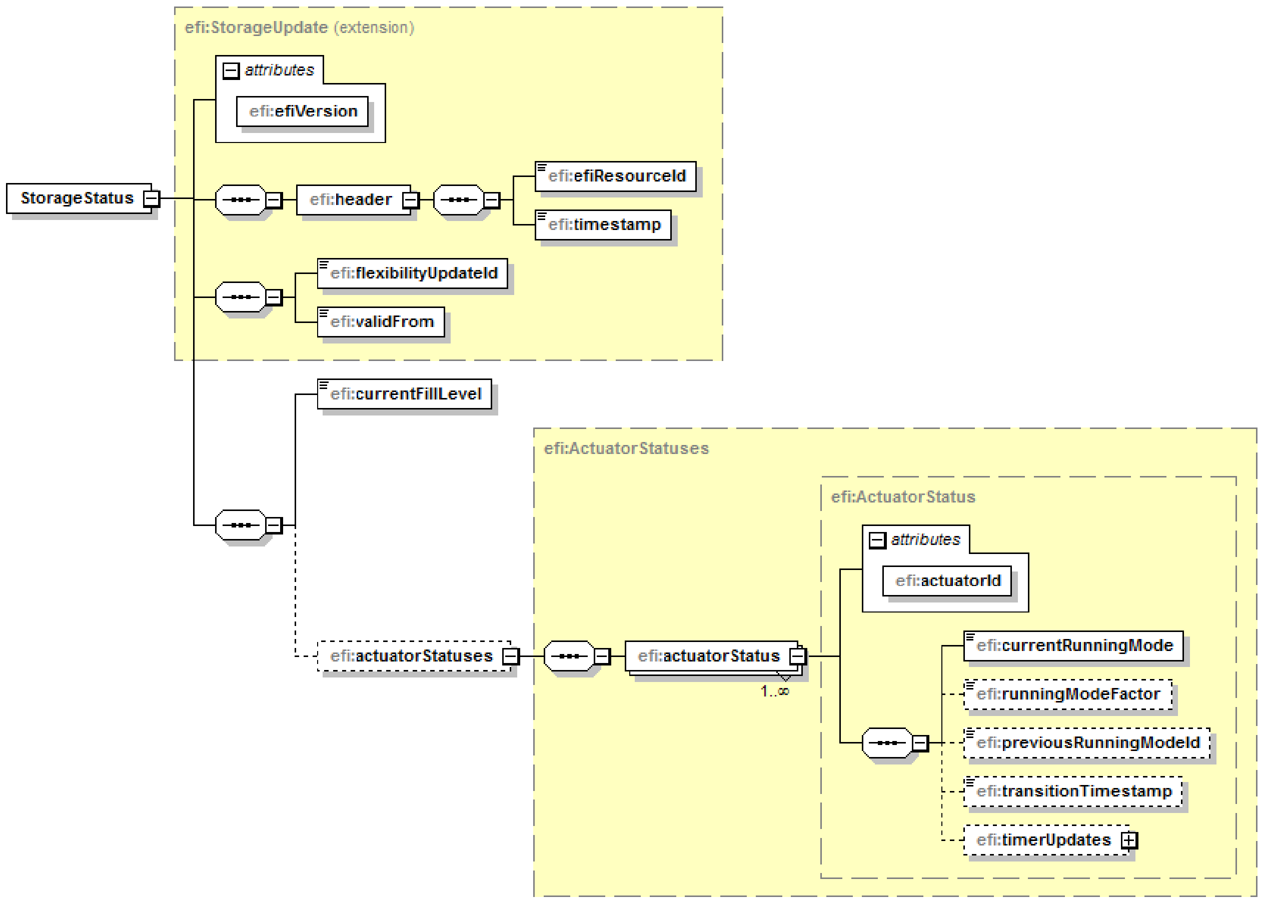

FIGURE 64: OVERVIEW OF THE STORAGESTATUS CLASS

The StorageStatus class is derived from the StorageUpdate class (see 7.3) and in addition features the following attributes:

Attribute	Description
currentFillLevel (double)	This attribute expresses the current fill level of the storage.
actuatorStatuses (ActuatorStatuses)	This optional attribute is of the type ActuatorStatuses and is explained in paragraph 7.5.1.

TABLE 60: ATTRIBUTES OF THE STORAGESTATUS CLASS

7.5.1 ActuatorStatuses

This class provides information about the current status of the actuators. It has the following attributes:

Attribute	Description
actuatorStatus (ActuatorStatus)	This attribute describes a single actuatorStatus. There can be multiple actuatorStatus attributes within the ActuatorStatuses class. The ActuatorStatus type is explained in paragraph 7.5.2.

TABLE 61: ATTRIBUTES OF THE ACTUATORSTATUSES CLASS

7.5.2 ActuatorStatus

This class provides the status of a single actuator, it has the following attributes:

Attribute	Description
actuatorId (int)	This is a unique ID within the context of this storage device to identify this actuator. It references the actuator that was declared in the StorageRegistration message.
currentRunningMode (int)	This is a unique ID within the context of this actuator to identify the currentRunningMode. It references one of the running modes that were declared in the StorageSystemDescription message.
runningModeFactor (double)	This attribute is optional, but MUST be used when the currentRunningModeId refers to a continuous running mode. The factor must be a number between 0 and 1. When it is 0 the lower bound value must be used for all involved commodites and the filling rate, the upper bound value must be used when this factor is 1. If the factor is between 0 and 1, then that portion of the range

	should be added to the lower bound values.
previousRunningModeId (int)	This attribute is optional, but MUST be provided by the Resource Manager when this information is available. It refers to the ID of the running mode that was selected before the current running mode.
transitionTimestamp (dateTime)	This attribute is optional, but MUST be provided by the Resource Manager when this information is available. It refers to the point in time when the switch to the current running mode was made. If applicable the CEM can combine this with the transitionDuration information provided in the StorageSystemDescription message to derive whether the device might still be ramping up or down.
timerUpdates (TimerUpdates)	The timerUpdates attribute is of the type TimerUpdates and is explained in 6.5.1.

TABLE 62: ATTRIBUTES OF THE ACTUATORUPDATE CLASS

7.6 StorageFillLevelTargetProfile

Normally the fill level in the storage only needs to stay between the lower and upper bound (provided in the system description message) to meet the constraints of the device. The end-user might also have additional requirements that have to be met.

Consider for an electrical vehicle for instance that returns almost empty from work at 18.00 o'clock in the evening. The end-user wants the vehicle to be charged for at least 20% by 19.00, so that she can always reach the nearest hospital in case of an emergency. At 7.30 in the morning the vehicle needs to be charged for at least 80% to be able to make the commute to work.

These additional constraints can be communicated by means of the StorageFillLevelTargetProfile message.

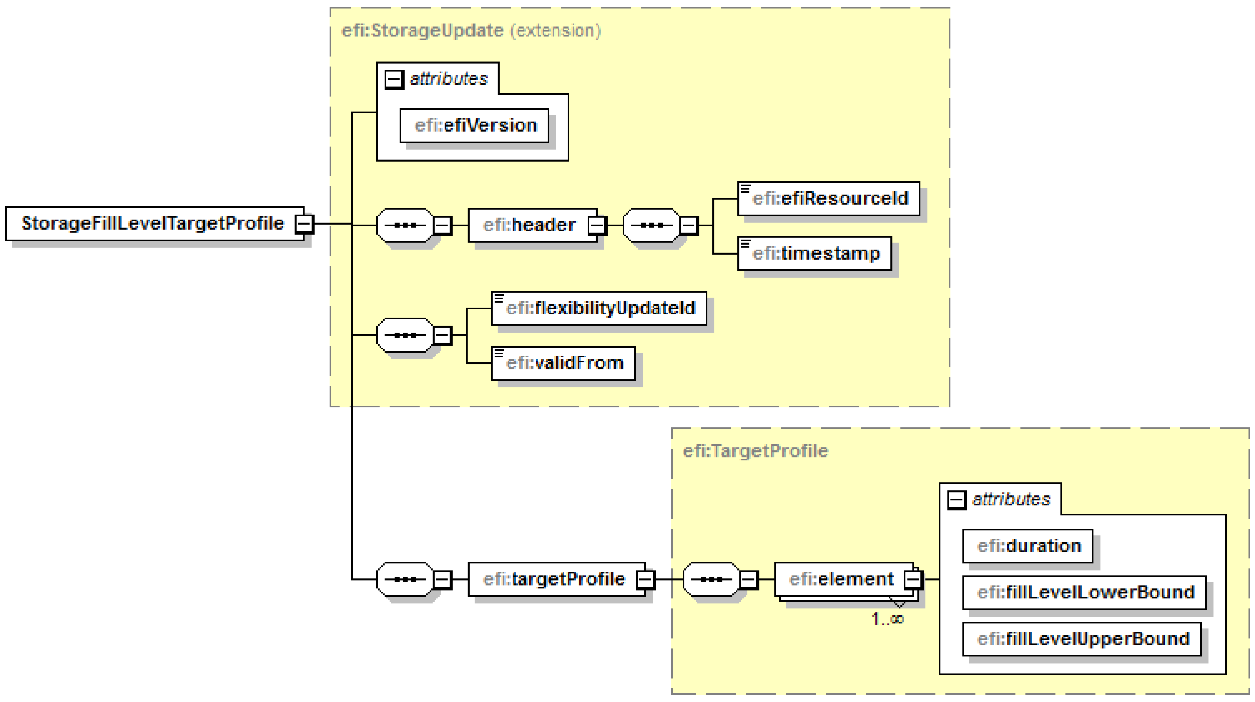

FIGURE 65: OVERVIEW OF THE STORAGEFILLLEVELTARGETPROFILE CLASS

The StorageFillLevelTargetProfile class is derived from the StorageUpdate class (see 7.3) and in addition features the following attributes:

Attribute	Description
targetProfile (TargetProfile)	This attribute contains the target profile for the fill level. The start time of this profile is equal to the validFrom attribute of the generic StorageUpdate class from which the StorageFillLevelTargetProfile is derived. This attribute is of the the type TargetProfile and is further explained in 7.6.1.

TABLE 63: ATTRIBUTES OF THE STORAGEFILLLEVELTARGETPROFILE CLASS

7.6.1 TargetProfile

The TargetProfile class consists of one or multiple elements that represent timeslots in the target profile. An element contains the following attributes:

Attribute	Description
duration (duration)	This attribute represents the duration of an element in the target profile.
fillLevelLowerBound (double)	This attribute represents the lower bound of the fill level for this element. The actual fill level during this timeslot

	MUST be greater or equal to this value.
fillLevelUpperBound (double)	This attribute represents the upper bound of the fill level for this element. The actual fill level during this timeslot MUST be smaller or equal to this value.

TABLE 64: ATTRIBUTES OF THE TARGETPROFILE CLASS

7.7 StorageUsageForecast

To be done.

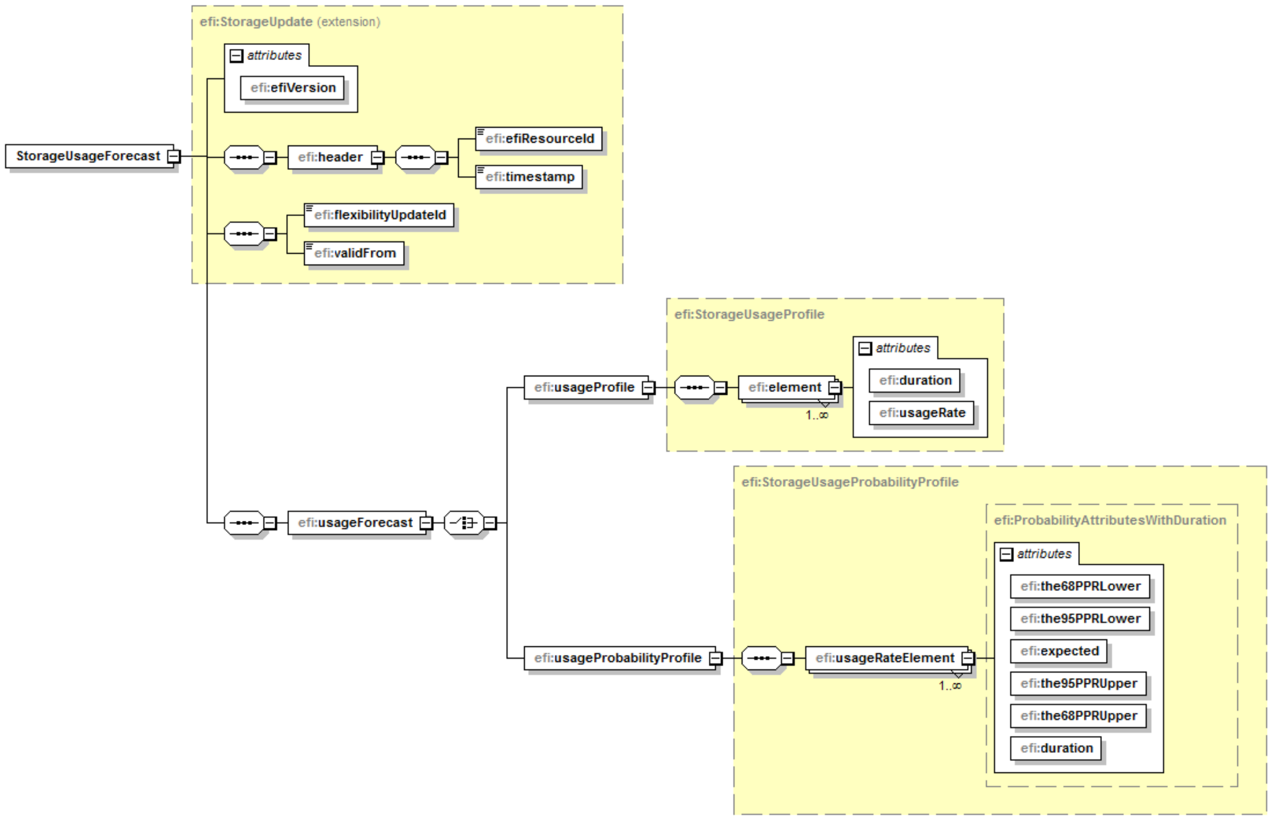

FIGURE 66: OVERVIEW OF THE STORAGEUSAGEFORECAST CLASS

7.8 StorageInstruction

This instruction message is being sent from the CEM to the RM. It contains the selection of a particular running mode as well as the desired switch time.

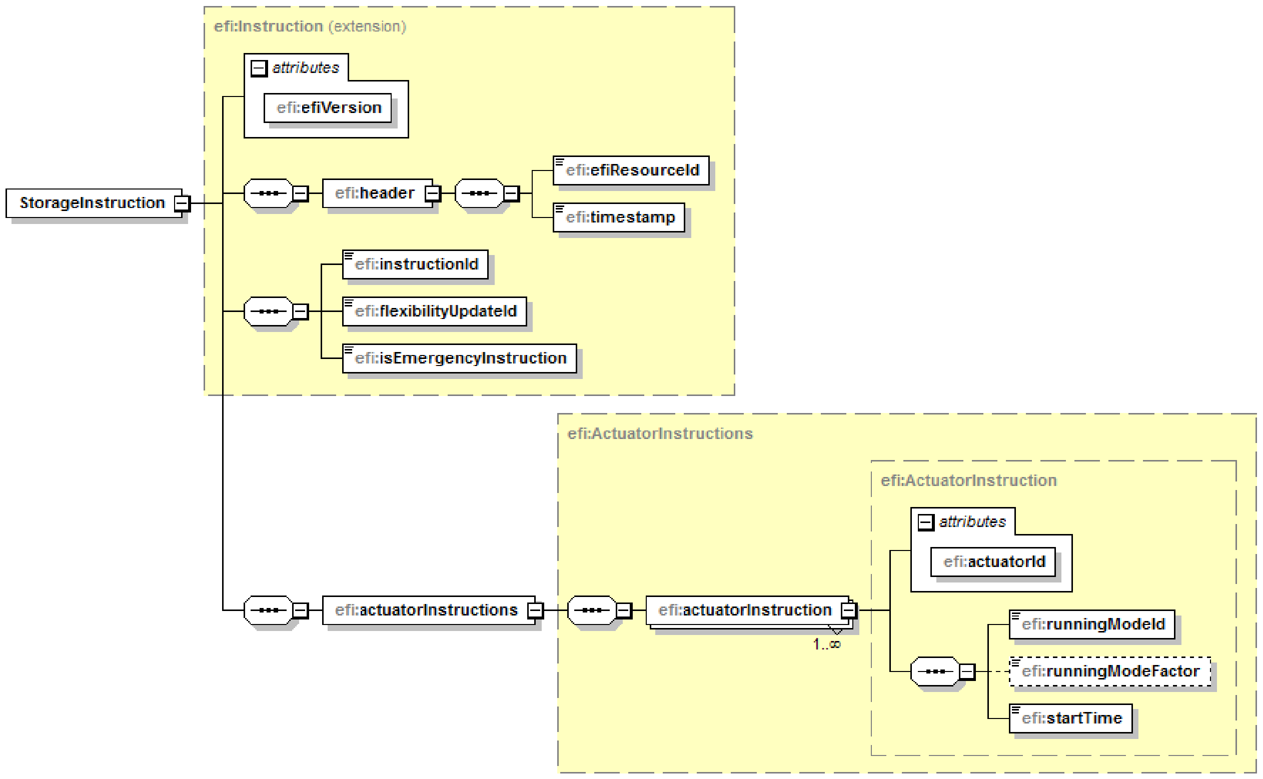

FIGURE 67: OVERVIEW OF THE STORAGEINSTRUCTION CLASS

The StorageInstruction class is derived from Instruction (see section 3.5) and in addition features the following attributes:

Attribute	Description
actuatorInstructions (ActuatorInstructions)	This attribute contains a list of actuator instructions. It is of the type ActuatorInstructions and is explained in paragraph 7.8.1.

TABLE 65: ATTRIBUTES OF THE STORAGEINSTRUCTION CLASS

7.8.1 ActuatorInstructions

This class provides a list of actuator instructions, it has the following attributes:

Attribute	Description
actuatorInstruction (ActuatorInstruction)	This attribute describes a single actuatorInstruction. There can be multiple actuatorInstruction attributes

	within the ActuatorInstructions class. The ActuatorInstruction type is explained in paragraph 7.8.2.

TABLE 66: ATTRIBUTES OF THE ACTUATORINSTRUCTIONS CLASS

7.8.2 ActuatorInstruction

This class contains the instruction for a single actuator, it has the following attributes:

Attribute	Description
actuatorId (int)	This is a unique ID within the context of this storage device to identify this actuator. It references the actuator that was declared in the StorageRegistration message.
runningModeId (int)	This is a unique ID within the context of this actuator to identify the running mode. It references one of the running modes that were declared in the StorageSystemDescription message.
runningModeFactor (double)	This optional attribute only needs to be provided when this ActuatorInstruction concerns a continuous running mode. In that case a factor (value between 0 and 1) has to be specified. It selects the desired values in the range between the lower and the upper bound.
startTime (dateTime)	The point in time at which this instruction should be executed by the RM.

TABLE 67: ATTRIBUTES OF THE ACTUATORINSTRUCTION CLASS

Annex A EFI XML schema

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one
  or more contributor license agreements.  See the NOTICE file
  distributed with this work for additional information
  regarding copyright ownership.  The ASF licenses this file
  to you under the Apache License, Version 2.0 (the
  "License"); you may not use this file except in compliance
  with the License.  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing,
  software distributed under the License is distributed on an
  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
  KIND, either express or implied.  See the License for the
  specific language governing permissions and limitations
  under the License.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:efi="http://www.flexiblepower.org/efi-2"
  targetNamespace="http://www.flexiblepower.org/efi-2" elementFormDefault="qualified"
  attributeFormDefault="qualified">
  <complexType name="EfiMessage" abstract="true">
    <sequence>
      <element name="header">
        <complexType>
          <sequence>
            <element name="efiResourceId" type="efi:Identifier" />
            <element name="timestamp" type="dateTime" />
          </sequence>
        </complexType>
      </element>
    </sequence>
    <attribute name="efiVersion" type="string" use="required" fixed="2.0" />
  </complexType>
  <complexType name="DeviceDescription">
    <sequence>
      <element name="deviceClass" type="efi:DeviceClass" minOccurs="1" maxOccurs="1" />
      <element name="serialNumber" type="string" minOccurs="0" maxOccurs="1" />
      <element name="label" type="string" minOccurs="0" maxOccurs="1" />
    </sequence>
  </complexType>
  <complexType name="FlexibilityRegistration">
    <complexContent>
      <extension base="efi:EfiMessage">
        <sequence>
          <element name="instructionProcessingDelay" type="duration" minOccurs="1"
            maxOccurs="1" />
          <element name="deviceDescription" type="efi:DeviceDescription" minOccurs="1"
            maxOccurs="1" />
          <element name="currency" type="efi:CurrencyType" minOccurs="0" maxOccurs="1" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="FlexibilityUpdate" abstract="true">
    <complexContent>
      <extension base="efi:EfiMessage">
        <sequence>
          <element name="flexibilityUpdateId" type="efi:Identifier" />
          <element name="validFrom" type="dateTime" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="Instruction" abstract="true">
```

```
    <complexContent>
      <extension base="efi:EfiMessage">
        <sequence>
          <element name="instructionId" type="efi:Identifier" />
          <element name="flexibilityUpdateId" type="efi:Identifier" />
          <element name="isEmergencyInstruction" type="boolean" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="InstructionStatusUpdate">
    <complexType>
      <complexContent>
        <extension base="efi:EfiMessage">
          <sequence>
            <element name="instructionId" type="efi:Identifier" />
            <element name="status" type="efi:InstructionStatus" />
            <element name="debugInformation" type="string" minOccurs="0" maxOccurs="1" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <simpleType name="InstructionStatus">
    <restriction base="string">
      <enumeration value="ACCEPTED" />
      <enumeration value="STARTED" />
      <enumeration value="SUCCEEDED" />
      <enumeration value="REJECTED" />
      <enumeration value="ABORTED" />
    </restriction>
  </simpleType>
  <element name="FlexibilityRevoke">
    <complexType>
      <complexContent>
        <extension base="efi:EfiMessage" />
      </complexContent>
    </complexType>
  </element>
  <element name="InstructionRevoke">
    <complexType>
      <complexContent>
        <extension base="efi:EfiMessage">
          <sequence>
            <element name="instructionId" type="efi:Identifier" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <simpleType name="CommodityEnum">
    <restriction base="string">
      <enumeration value="ELECTRICITY" />
      <enumeration value="GAS" />
      <enumeration value="HEAT" />
    </restriction>
  </simpleType>
  <simpleType name="CurtailmentQuantity">
    <restriction base="string">
      <enumeration value="ELECTRICITY.POWER" />
      <enumeration value="GAS.FLOWRATE" />
      <enumeration value="HEAT.TEMPERATURE" />
      <enumeration value="HEAT.FLOWRATE" />
      <enumeration value="HEAT.THERMALPOWER" />
    </restriction>
  </simpleType>
  <complexType name="ProbabilityAttributes">
    <attribute name="the68PPRLower" type="double" use="required" />
    <attribute name="the95PPRLower" type="double" use="required" />
    <attribute name="expected" type="double" use="required" />
    <attribute name="the95PPRUpper" type="double" use="required" />
    <attribute name="the68PPRUpper" type="double" use="required" />
  </complexType>
  <complexType name="ProbabilityAttributesWithDuration">
    <complexContent>
      <extension base="efi:ProbabilityAttributes">
        <attribute name="duration" type="duration" use="required" />
      </extension>
    </complexContent>
  </complexType>
  <complexType name="StorageUsageProfile">
    <sequence>
      <element name="element" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="duration" type="duration" use="required" />
          <attribute name="usage" type="double" use="required" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <complexType name="StorageUsageProbabilityProfile">
    <sequence>
      <element name="usageElement" minOccurs="1" maxOccurs="unbounded"
        type="efi:ProbabilityAttributesWithDuration" />
    </sequence>
  </complexType>
  <complexType name="ElectricityProfile">
    <sequence>
      <element name="element" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="duration" type="duration" use="required" />
          <attribute name="power" type="double" use="required" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <complexType name="ElectricityProbabilityProfile">
    <sequence>
      <element name="powerElement" minOccurs="1" maxOccurs="unbounded"
        type="efi:ProbabilityAttributesWithDuration" />
    </sequence>
  </complexType>
  <complexType name="GasProfile">
    <sequence>
      <element name="element" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="duration" type="duration" use="required" />
          <attribute name="flowRate" type="double" use="required" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <complexType name="GasProbabilityProfile">
    <sequence>
      <element name="flowRateElement" minOccurs="1" maxOccurs="unbounded"
        type="efi:ProbabilityAttributesWithDuration" />
    </sequence>
  </complexType>
  <complexType name="HeatProfile">
    <sequence>
      <element name="element" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="duration" type="duration" use="required" />
          <attribute name="temperature" type="double" use="optional" />
          <attribute name="flowRate" type="double" use="optional" />
          <attribute name="thermalPower" type="double" use="optional" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <complexType name="HeatProbabilityProfile">
    <sequence>
      <element name="element" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <sequence>
            <element name="temperature" minOccurs="0" maxOccurs="1" type="efi:ProbabilityAttributes" />
            <element name="flowRate" minOccurs="0" maxOccurs="1" type="efi:ProbabilityAttributes" />
            <element name="thermalPower" minOccurs="0" maxOccurs="1" type="efi:ProbabilityAttributes"
/>
          </sequence>
          <attribute name="duration" type="duration" use="required" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <element name="InflexibleRegistration">
    <complexType>
      <complexContent>
        <extension base="efi:FlexibilityRegistration">
          <sequence>
            <element name="supportedCommodities" type="efi:SupportedCommodities" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="SupportedCommodities">
    <sequence>
      <element name="commodityType" type="efi:CommodityEnum" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="CurtailmentOptions">
    <sequence>
      <element name="curtailmentOption" type="efi:CurtailmentOption" minOccurs="0"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="CurtailmentOption">
    <sequence>
      <element name="curtailmentRange" type="efi:CurtailmentRange" minOccurs="1" maxOccurs="unbounded"
/>
    </sequence>
    <attribute name="curtailmentQuantity" type="efi:CurtailmentQuantity" use="required" />
    <attribute name="minimalCurtailmentDuration" type="duration" use="optional" default="PT0S" />
  </complexType>
  <complexType name="CurtailmentRange">
    <attribute name="lowerBound" type="double" use="required" />
    <attribute name="upperBound" type="double" use="required" />
  </complexType>
  <complexType name="InflexibleUpdate">
    <complexContent>
      <extension base="efi:FlexibilityUpdate" />
    </complexContent>
  </complexType>
  <element name="Measurement">
    <complexType>
      <complexContent>
        <extension base="efi:EfiMessage">
          <sequence>
            <element name="measurementTimestamp" type="dateTime" />
            <element name="electricityMeasurement" minOccurs="0" maxOccurs="1">
              <complexType>
                <attribute name="power" type="double" use="required" />
              </complexType>
            </element>
            <element name="gasMeasurement" minOccurs="0" maxOccurs="1">
              <complexType>
                <attribute name="flowRate" type="double" use="required" />
              </complexType>
            </element>
            <element name="heatMeasurement" minOccurs="0" maxOccurs="1">
              <complexType>
                <attribute name="temperature" type="double" use="optional" />
                <attribute name="flowRate" type="double" use="optional" />
                <attribute name="thermalPower" type="double" use="optional" />
              </complexType>
            </element>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <element name="InflexibleForecast" abstract="true">
    <complexType>
      <complexContent>
        <extension base="efi:InflexibleUpdate">
          <sequence>
            <element name="forecastProfiles" type="efi:ProfileContainer" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="ProfileContainer">
    <sequence>
      <choice minOccurs="0" maxOccurs="1">
        <element name="electricityProfile" type="efi:ElectricityProfile" />
        <element name="electricityProbabilityProfile" type="efi:ElectricityProbabilityProfile" />
      </choice>
      <choice minOccurs="0" maxOccurs="1">
        <element name="gasProfile" type="efi:GasProfile" />
        <element name="gasProbabilityProfile" type="efi:GasProbabilityProfile" />
      </choice>
      <choice minOccurs="0" maxOccurs="1">
        <element name="heatProfile" type="efi:HeatProfile" />
        <element name="heatProbabilityProfile" type="efi:HeatProbabilityProfile" />
      </choice>
    </sequence>
  </complexType>
  <element name="InflexibleCurtailmentOptions">
    <complexType>
      <complexContent>
        <extension base="efi:InflexibleUpdate">
          <sequence>
            <element name="curtailmentOptions" type="efi:CurtailmentOptions" minOccurs="0" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <element name="InflexibleInstruction">
    <complexType>
      <complexContent>
        <extension base="efi:Instruction">
          <sequence>
            <element name="curtailmentProfile" type="efi:CurtailmentProfile" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="CurtailmentProfile">
    <sequence>
      <element name="curtailmentProfileElement" type="efi:CurtailmentProfileElement"
        minOccurs="1" maxOccurs="unbounded" />
    </sequence>
```

```
    <attribute name="curtailmentQuantity" type="efi:CurtailmentQuantity" use="required" />
    <attribute name="startTime" type="dateTime" use="required" />
  </complexType>
  <complexType name="CurtailmentProfileElement">
    <attribute name="duration" type="duration" />
    <attribute name="value" type="double" />
  </complexType>
  <element name="ShiftableRegistration">
    <complexType>
      <complexContent>
        <extension base="efi:FlexibilityRegistration">
          <sequence>
            <element name="supportedCommodities" type="efi:SupportedCommodities" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="SequentialProfile">
    <sequence>
      <element name="maxIntervalBefore" type="duration" />
      <element name="sequentialProfileAlternatives" type="efi:SequentialProfileAlternatives"
        minOccurs="1" maxOccurs="unbounded" />
    </sequence>
    <attribute name="sequenceNr" type="int" use="required" />
  </complexType>
  <complexType name="SequentialProfileAlternative">
    <complexContent>
      <extension base="efi:ProfileContainer">
        <attribute name="alternativeNr" type="int" use="required" />
      </extension>
    </complexContent>
  </complexType>
  <complexType name="SequentialProfileAlternatives">
    <sequence>
      <element name="sequentialProfileAlternative" type="efi:SequentialProfileAlternative" />
    </sequence>
  </complexType>
  <complexType name="SequentialProfiles">
    <sequence>
      <element name="sequentialProfile" type="efi:SequentialProfile" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <element name="ShiftableUpdate">
    <complexType>
      <complexContent>
        <extension base="efi:FlexibilityUpdate">
          <sequence>
            <element name="endBefore" type="dateTime" />
            <element name="sequentialProfiles" type="efi:SequentialProfiles" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="SequentialProfileInstruction">
    <sequence>
      <element name="sequenceNr" type="int" />
      <element name="alternativeNr" type="int" />
      <element name="startTime" type="dateTime" />
    </sequence>
  </complexType>
  <complexType name="SequentialProfileInstructions">
    <sequence>
      <element name="sequentialProfileInstruction" type="efi:SequentialProfileInstruction"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <element name="ShiftableInstruction">
```

```
    <complexType>
      <complexContent>
        <extension base="efi:Instruction">
          <sequence>
            <element name="sequentialProfileInstructions" type="efi:SequentialProfileInstructions"
              minOccurs="1" maxOccurs="1" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <element name="StorageRegistration">
    <complexType>
      <complexContent>
        <extension base="efi:FlexibilityRegistration">
          <sequence>
            <element name="fillLevelLabel" type="string" />
            <element name="fillLevelUnit" type="string" />
            <element name="actuators" type="efi:Actuators" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="Actuator">
    <sequence>
      <element name="supportedCommodity" type="efi:CommodityEnum" maxOccurs="unbounded" />
    </sequence>
    <attribute name="id" type="int" use="required" />
    <attribute name="label" type="string" use="optional" />
  </complexType>
  <complexType name="Actuators">
    <sequence>
      <element name="actuator" type="efi:Actuator" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="Timer">
    <attribute name="id" type="int" use="required" />
    <attribute name="label" type="string" use="required" />
    <attribute name="duration" type="duration" default="PT0S" />
  </complexType>
  <complexType name="Timers">
    <sequence>
      <element name="timer" type="efi:Timer" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="Transitions">
    <sequence>
      <element name="transition" type="efi:Transition" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="TimerReferences">
    <sequence>
      <element name="timerReference" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="timerId" type="int" use="required" />
        </complexType>
      </element>
    </sequence>
  </complexType>
  <complexType name="Transition">
    <sequence>
      <element name="startTimers" type="efi:TimerReferences" minOccurs="0" maxOccurs="1" />
      <element name="blockingTimers" type="efi:TimerReferences" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attribute name="fromRunningModeId" type="int" use="required" />
    <attribute name="toRunningModeId" type="int" use="required" />
    <attribute name="transitionCost" type="double" />
    <attribute name="transitionTime" type="duration" use="optional" />
  </complexType>
  <simpleType name="CurrencyType">
    <restriction base="string">
      <enumeration value="AED" />
      <enumeration value="ANG" />
      <enumeration value="AUD" />
      <enumeration value="CHE" />
      <enumeration value="CHF" />
      <enumeration value="CHW" />
      <enumeration value="EUR" />
      <enumeration value="GBP" />
      <enumeration value="LBP" />
      <enumeration value="LKR" />
      <enumeration value="LRD" />
      <enumeration value="LSL" />
      <enumeration value="LYD" />
      <enumeration value="MAD" />
      <enumeration value="MDL" />
      <enumeration value="MGA" />
      <enumeration value="MKD" />
      <enumeration value="MMK" />
      <enumeration value="MNT" />
      <enumeration value="MOP" />
      <enumeration value="MRO" />
      <enumeration value="MUR" />
      <enumeration value="MVR" />
      <enumeration value="MWK" />
      <enumeration value="MXN" />
      <enumeration value="MXV" />
      <enumeration value="MYR" />
      <enumeration value="MZN" />
      <enumeration value="NAD" />
      <enumeration value="NGN" />
      <enumeration value="NIO" />
      <enumeration value="NOK" />
      <enumeration value="NPR" />
      <enumeration value="NZD" />
      <enumeration value="OMR" />
      <enumeration value="PAB" />
      <enumeration value="PEN" />
      <enumeration value="PGK" />
      <enumeration value="PHP" />
      <enumeration value="PKR" />
      <enumeration value="PLN" />
      <enumeration value="PYG" />
      <enumeration value="QAR" />
      <enumeration value="RON" />
      <enumeration value="RSD" />
      <enumeration value="RUB" />
      <enumeration value="RWF" />
      <enumeration value="SAR" />
      <enumeration value="SBD" />
      <enumeration value="SCR" />
      <enumeration value="SDG" />
      <enumeration value="SEK" />
      <enumeration value="SGD" />
      <enumeration value="SHP" />
      <enumeration value="SLL" />
      <enumeration value="SOS" />
      <enumeration value="SRD" />
      <enumeration value="SSP" />
      <enumeration value="STD" />
      <enumeration value="SYP" />
      <enumeration value="SZL" />
      <enumeration value="THB" />
      <enumeration value="TJS" />
      <enumeration value="TMT" />
      <enumeration value="TND" />
      <enumeration value="TOP" />
      <enumeration value="TRY" />
      <enumeration value="TTD" />
      <enumeration value="TWD" />
      <enumeration value="TZS" />
      <enumeration value="UAH" />
      <enumeration value="UGX" />
      <enumeration value="USD" />
      <enumeration value="USN" />
      <enumeration value="UYI" />
      <enumeration value="UYU" />
      <enumeration value="UZS" />
      <enumeration value="VEF" />
      <enumeration value="VND" />
      <enumeration value="VUV" />
      <enumeration value="WST" />
      <enumeration value="XAG" />
      <enumeration value="XAU" />
      <enumeration value="XBA" />
      <enumeration value="XBB" />
      <enumeration value="XBC" />
      <enumeration value="XBD" />
      <enumeration value="XCD" />
      <enumeration value="XOF" />
      <enumeration value="XPD" />
      <enumeration value="XPF" />
      <enumeration value="XPT" />
      <enumeration value="XSU" />
      <enumeration value="XTS" />
      <enumeration value="XUA" />
      <enumeration value="XXX" />
      <enumeration value="YER" />
      <enumeration value="ZAR" />
      <enumeration value="ZMW" />
      <enumeration value="ZWL" />
    </restriction>
  </simpleType>
  <complexType name="RunningMode" abstract="true">
    <attribute name="id" type="int" use="required" />
    <attribute name="label" type="string" use="required" />
  </complexType>
  <complexType name="StorageRunningModeElement" abstract="true">
    <attribute name="fillLevelLowerBound" type="double" use="required" />
    <attribute name="fillLevelUpperBound" type="double" use="required" />
  </complexType>
  <complexType name="StorageDiscreteRunningMode">
    <complexContent>
      <extension base="efi:RunningMode">
        <sequence>
          <element name="discreteRunningModeElement" minOccurs="1" maxOccurs="unbounded">
            <complexType>
              <complexContent>
                <extension base="efi:StorageRunningModeElement">
                  <sequence>
                    <element name="fillingRate" type="double" minOccurs="1" maxOccurs="1" />
                    <element name="runningCost" type="decimal" minOccurs="0" maxOccurs="1" />
                    <element name="electricityPower" type="double" minOccurs="0" maxOccurs="1" />
                    <element name="gasFlawRate" type="double" minOccurs="0" maxOccurs="1" />
                    <element name="heatTemperature" type="double" minOccurs="0" maxOccurs="1" />
                    <element name="heatFlowRate" type="double" minOccurs="0" maxOccurs="1" />
                    <element name="heatThermalPower" type="double" minOccurs="0" maxOccurs="1" />
                  </sequence>
                </extension>
              </complexContent>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

```
<complexType name="StorageContinuousRunningModeData" abstract="true">
  <sequence>
    <element name="fillingRate" type="double" minOccurs="1" />
    <element name="runningCost" type="decimal" minOccurs="0" />
    <element name="electricityPower" type="double" minOccurs="0" maxOccurs="1" />
    <element name="gasFlawRate" type="double" minOccurs="0" maxOccurs="1" />
    <element name="heatTemperature" type="double" minOccurs="0" maxOccurs="1" />
    <element name="heatFlowRate" type="double" minOccurs="0" maxOccurs="1" />
    <element name="heatThermalPower" type="double" minOccurs="0" maxOccurs="1" />
  </sequence>
</complexType>
<complexType name="StorageContinuousRunningMode">
  <complexContent>
    <extension base="efi:RunningMode">
      <sequence>
        <element name="continuousRunningModeElement" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <extension base="efi:StorageRunningModeElement">
                <sequence>
                  <element name="lowerBound">
                    <complexType>
                      <complexContent>
                        <extension base="efi:StorageContinuousRunningModeData" />
                      </complexContent>
                    </complexType>
                  </element>
                  <element name="upperBound">
                    <complexType>
                      <complexContent>
                        <extension base="efi:StorageContinuousRunningModeData" />
                      </complexContent>
                    </complexType>
                  </element>
                </sequence>
              </extension>
            </complexContent>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="StorageRunningModes">
  <sequence>
    <choice minOccurs="1" maxOccurs="unbounded">
      <element name="discreteRunningMode" type="efi:StorageDiscreteRunningMode" />
      <element name="continuousRunningMode" type="efi:StorageContinuousRunningMode" />
    </choice>
  </sequence>
</complexType>
<complexType name="ActuatorBehaviour">
  <sequence>
    <element name="runningModes" type="efi:StorageRunningModes" />
    <element name="timers" type="efi:Timers" minOccurs="0" maxOccurs="1" />
    <element name="transitions" type="efi:Transitions" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attribute name="actuatorId" type="string" use="required" />
</complexType>
<complexType name="LeakageElement">
  <attribute name="fillLevelLowerBound" type="double" use="required" />
  <attribute name="fillLevelUpperBound" type="double" use="required" />
  <attribute name="leakageRate" type="double" use="required" />
</complexType>
<complexType name="LeakageFunction">
  <sequence>
    <element name="leakageElement" type="efi:LeakageElement" minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>
```

```
<complexType name="ActuatorBehaviours">
  <sequence>
    <element name="actuatorBehaviour" type="efi:ActuatorBehaviour" minOccurs="1"
      maxOccurs="unbounded" />
  </sequence>
</complexType>
<element name="StorageSystemDescription">
  <complexType>
    <complexContent>
      <extension base="efi:StorageUpdate">
        <sequence>
          <element name="actuatorBehaviours" type="efi:ActuatorBehaviours" minOccurs="0"
            maxOccurs="1" />
          <element name="leakageBehaviour" type="efi:LeakageFunction" minOccurs="0"
            maxOccurs="1" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
<complexType name="StorageUpdate" abstract="true">
  <complexContent>
    <extension base="efi:FlexibilityUpdate" />
  </complexContent>
</complexType>
<complexType name="TimerUpdate">
  <sequence>
    <element name="finishedAt" type="dateTime" />
  </sequence>
  <attribute name="timerId" type="int" use="required" />
</complexType>
<complexType name="TimerUpdates">
  <sequence>
    <element name="timerUpdate" type="efi:TimerUpdate" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
<complexType name="ActuatorUpdate">
  <sequence>
    <element name="currentRunningMode" type="int" />
    <element name="timerUpdates" type="efi:TimerUpdates" minOccurs="0" />
  </sequence>
  <attribute name="actuatorId" type="int" use="required" />
</complexType>
<complexType name="ActuatorUpdates">
  <sequence>
    <element name="actuatorUpdate" type="efi:ActuatorUpdate" minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>
<element name="StorageStateUpdate">
  <complexType>
    <complexContent>
      <extension base="efi:StorageUpdate">
        <sequence>
          <element name="currentFillLevel" type="double" />
          <element name="actuatorUpdates" type="efi:ActuatorUpdates" minOccurs="0" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
<complexType name="TargetProfile">
  <sequence>
    <element name="element" maxOccurs="unbounded">
      <complexType>
        <attribute name="duration" type="duration" use="required" />
        <attribute name="fillLevelLowerBound" type="double" use="required" />
        <attribute name="fillLevelUpperBound" type="double" use="required" />
      </complexType>
    </element>
    </sequence>
    <attribute name="startTime" type="dateTime" />
  </complexType>
  <element name="StorageTargetProfileUpdate">
    <complexType>
      <complexContent>
        <extension base="efi:StorageUpdate">
          <sequence>
            <element name="targetProfile" type="efi:TargetProfile" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <element name="StorageUsageForecastUpdate">
    <complexType>
      <complexContent>
        <extension base="efi:StorageUpdate">
          <sequence>
            <element name="storageUsageForecast">
              <complexType>
                <choice>
                  <element name="storageUsageProfile" type="efi:StorageUsageProfile" />
                  <element name="storageUsageProbabilityProfile"
type="efi:StorageUsageProbabilityProfile" />
                </choice>
              </complexType>
            </element>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="ActuatorInstruction">
    <sequence>
      <element name="runningModeId" type="int" />
      <element name="runningModeFactor" minOccurs="0" maxOccurs="1">
        <simpleType>
          <restriction base="double">
            <minInclusive value="0.0" />
            <maxInclusive value="1.0" />
          </restriction>
        </simpleType>
      </element>
      <element name="startTime" type="dateTime" />
    </sequence>
    <attribute name="actuatorId" type="int" use="required" />
  </complexType>
  <complexType name="ActuatorInstructions">
    <sequence>
      <element name="actuatorInstruction" type="efi:ActuatorInstruction" minOccurs="1"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <element name="StorageInstruction">
    <complexType>
      <complexContent>
        <extension base="efi:Instruction">
          <sequence>
            <element name="actuatorInstructions" type="efi:ActuatorInstructions" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <element name="AdjustableRegistration">
    <complexType>
      <complexContent>
        <extension base="efi:FlexibilityRegistration">
```

```
          <sequence>
            <element name="supportedCommodities" type="efi:SupportedCommodities" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
  <complexType name="AdjustableDiscreteRunningMode">
    <complexContent>
      <extension base="efi:RunningMode">
        <sequence>
          <element name="runningCost" type="decimal" minOccurs="0" maxOccurs="1" />
          <element name="electricityConsumption" type="double" minOccurs="0" maxOccurs="1" />
          <element name="gasConsumption" type="double" minOccurs="0" maxOccurs="1" />
          <element name="heatConsumption" type="double" minOccurs="0" maxOccurs="1" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AdjustableContinuousRunningModeData" abstract="true">
    <sequence>
      <element name="runningCost" type="decimal" minOccurs="0" />
      <element name="electricityConsumption" type="double" minOccurs="0" />
      <element name="gasConsumption" type="double" minOccurs="0" />
      <element name="heatConsumption" type="double" minOccurs="0" />
    </sequence>
  </complexType>
  <complexType name="AdjustableContinuousRunningMode">
    <complexContent>
      <extension base="efi:RunningMode">
        <sequence>
          <element name="lowerBound">
            <complexType>
              <complexContent>
                <extension base="efi:AdjustableContinuousRunningModeData" />
              </complexContent>
            </complexType>
          </element>
          <element name="upperBound">
            <complexType>
              <complexContent>
                <extension base="efi:AdjustableContinuousRunningModeData" />
              </complexContent>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AdjustableRunningModes">
    <sequence>
      <choice minOccurs="1" maxOccurs="unbounded">
        <element name="discreteRunningMode" type="efi:AdjustableDiscreteRunningMode" />
        <element name="continuousRunningMode" type="efi:AdjustableContinuousRunningMode" />
      </choice>
    </sequence>
  </complexType>
  <element name="AdjustableSystemDescription">
    <complexType>
      <complexContent>
        <extension base="efi:AdjustableUpdate">
          <sequence>
            <element name="runningModes" type="efi:AdjustableRunningModes" />
            <element name="timers" type="efi:Timers" minOccurs="0" maxOccurs="1" />
            <element name="transitions" type="efi:Transitions" minOccurs="0" maxOccurs="1" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
```

```
</element>
<complexType name="AdjustableUpdate">
  <complexContent>
    <extension base="efi:FlexibilityUpdate" />
  </complexContent>
</complexType>
<element name="AdjustableStateUpdate">
  <complexType>
    <complexContent>
      <extension base="efi:AdjustableUpdate">
        <sequence>
          <element name="currentRunningModeId" type="int" />
          <element name="timerUpdates" type="efi:TimerUpdates" minOccurs="1" maxOccurs="1" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
<element name="AdjustableInstruction">
  <complexType>
    <complexContent>
      <extension base="efi:Instruction">
        <sequence>
          <element name="runningModeId" type="int" />
          <element name="runningModeFactor" minOccurs="0" maxOccurs="1">
            <simpleType>
              <restriction base="double">
                <minInclusive value="0.0" />
                <maxInclusive value="1.0" />
              </restriction>
            </simpleType>
          </element>
          <element name="startTime" type="dateTime" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
<simpleType name="DeviceClass">
  <restriction base="string">
    <enumeration value="Refrigerator" />
    <enumeration value="Freezer" />
    <enumeration value="Water Cooler" />
    <enumeration value="Water Heater" />
    <enumeration value="Washing Machine" />
    <enumeration value="Clothes Dryer" />
    <enumeration value="Combo Washer Dryer" />
    <enumeration value="Drying Cabinet" />
    <enumeration value="Dishwasher" />
    <enumeration value="Heatpump" />
    <enumeration value="Micro-CHP" />
    <enumeration value="Stationary Battery" />
    <enumeration value="Electrical Vehicle" />
    <enumeration value="PV Panel" />
    <enumeration value="Windmill" />
    <enumeration value="Solar Collector" />
    <enumeration value="Air Conditioner" />
    <enumeration value="Ventilation" />
    <enumeration value="Air Quality Appliance" />
    <enumeration value="Gas Geater" />
    <enumeration value="Floor Heating" />
    <enumeration value="Generator" />
    <enumeration value="Miscellaneous" />
  </restriction>
</simpleType>
<simpleType name="Identifier">
  <restriction base="string">
    <pattern value="[a-zA-Z0-9\-_:]{2,64}" />
  </restriction>
  </simpleType>
</schema>
```